

Lagrangian Duality based Algorithms in Online Scheduling

Nguyen Kim Thang*

IBISC, University of Evry Val d'Essonne, France

Abstract

We consider Lagrangian duality based approaches to design and analyze algorithms for online energy-efficient scheduling. First, we present a primal-dual framework. Our approach makes use of the Lagrangian weak duality and convexity to derive dual programs for problems which could be formulated as convex assignment problems. The duals have intuitive structures as the ones in linear programming. The constraints of the duals explicitly indicate the online decisions and naturally lead to competitive algorithms. Second, we use a dual-fitting approach, which also based on the weak duality, to study problems which are unlikely to admit convex relaxations. Through the analysis, we show an interesting feature in which primal-dual gives idea for designing algorithms while the analysis is done by dual-fitting.

We illustrate the advantages and the flexibility of the approaches through problems in different setting: from single machine to unrelated machine environments, from typical competitive analysis to the one with resource augmentation, from convex relaxations to non-convex relaxations.

*Research supported by FMJH program Gaspard Monge in Optimization and Operations Research and by EDF.

Contents

1	Introduction	2
1.1	Approaches and Contribution	3
1.2	Related work	6
1.3	Organization	7
2	Framework for Online Convex Assignment	7
3	Minimizing Total Energy plus Lost Values	9
4	Maximizing the Total Value minus Energy	12
5	Energy Minimization in Speed Scaling with Power Down Model	16
5.1	Speed Scaling without Wake-Up Cost	16
5.2	Speed Scaling with Wake-Up Cost	17
6	Minimizing Energy plus Weighted Flow-Time in Speed Scaling with Power Down Model	20
6.1	The Algorithm.	21
6.2	Analysis	22

1 Introduction

In the online setting, items arrive over time and one must determine how to serve items in order to optimize a quality of service without the knowledge about future. A popular measure for studying the performance of online algorithms is *competitive ratio* in the model of the worst-case analysis. An algorithm is said to be *c-competitive* if for any instance its objective value is within factor c of the optimal offline algorithm's one. Moreover, to remedy the limitation of pathological instances in the worst-case analysis, there is other model called *resource augmentation* [17]. In the latter, online algorithms are given an extra power and are compared to the optimal offline algorithm without that additional resource. This model has successfully provided theoretical evidence for heuristics with good performance in practice, especially in online scheduling where jobs arrive online and need to be processed on machines. We say a scheduling algorithm is *s-speed c-competitive* if for any input instance the objective value of the algorithm with machines of speed s is at most c times the objective value of the optimal offline scheduler with unit speed machines.

The most successful tool until now to analyze online algorithms is the potential function method. Potential functions have been designed to show that the corresponding algorithms behave well in an amortized sense. However, designing such potential functions is far from trivial and often yields little insight about how to design such potential functions and algorithms for related problems.

Recently, interesting approaches [3, 12, 20] based on mathematical programming have been presented in the search for a principled tool to design and analyze online scheduling algorithms. The approaches give insight about the nature of many scheduling problems, hence lead to algorithms which are usually simple and competitive.

1.1 Approaches and Contribution

In this paper, we present approaches based on Lagrangian duality in designing and analyzing algorithms for online energy-efficient scheduling problems.

Primal-Dual Approach. We first show a primal-dual framework for a general online convex assignment problem and its applications to online scheduling. In the framework, the algorithm decisions are interactively guided by the dual variables in the primal-dual sense.

The online convex assignment problem consists of a set of agents and a set of items which arrive over time. At the arrival of item j , the item needs to be (fractionally) assigned to some agents i . Let x_{ij} is the amount of j assigned to i . The problem is to minimize $\sum_i f_i(\sum_j a_{ij}x_{ij})$ under some constraints $g_i(\sum_j b_{ij}x_{ij}) \leq 0$ and $h_j(\sum_i c_{ij}x_{ij}) \leq 0$ for every i, j where functions f_i, g_i, h_j 's are convex. In offline setting, the optimal solutions are completely characterized by the KKT conditions (see [7] for example). However, for online setting, the conditions could not be satisfied due to the lack of knowledge about the future.

Our approach is the following. We first consider the problem as a primal convex mathematical program. Then we derive a Lagrangian dual program by the standard Lagrangian duality. Instead of analyzing directly the corresponding Lagrangian functions where in general one can not disentangle the objective and the constraints as well as the primal and dual variables, we exploit the convexity property of given functions and construct a dual program. In the latter, dual variables are separated from the primal ones. The construction is shown in Section 2. As the price of the separation procedure, the strong duality property is not guaranteed. However, the weak duality always holds and that is crucial (and sufficient) to deduce a lower bound for the given problem. The dual construction is not standard in optimization but the goal is to derive duals with intuitive structures similar to the ones in linear programming which are easier to work with. An advantage of the approach lies in the dual program in which the constraints could be maintained online. Moreover, the dual constraints explicitly indicate the online decisions and naturally lead to a competitive algorithm in the primal-dual sense.

The dual construction is inspired by [11] which used the primal-dual approach for online matching with concave return. In fact, Devanur and Jain [11] considered a matching problem with convex objective function and *linear* constraints. They linearized the objective function and derived the dual in the sense of linear programming. In our framework, we consider problems with convex objective and *convex* constraints¹. We then construct our duals from Lagrangian dual programs. Informally, the construction could be seen as a linearization of Lagrangian duals.

Applications. We illustrate the advantages and the flexibility of the approach through online scheduling problems related to throughput. In the setting, there are a set of unrelated machines. Each job j is released at time r_j , has deadline d_j , a value a_j and a processing volume p_{ij} if job j is executed in machine i . Jobs could be executed preemptively but migration is not allowed, i.e., no job could be executed in more than one machine. At any time t , the scheduler has to choose an assignment of jobs to machines and the speed of each machine in order to process such jobs. The energy cost of machine i is $\int_0^\infty P(s_i(t))dt$ where P is a given convex energy power and $s_i(t)$ is the speed of machine i at time t . Typically, $P(z) = z^\alpha$ for some constant $\alpha \geq 1$. In the setting, we look for competitive and energy-efficient algorithms. The following objectives are natural ones

¹The primal-dual machinery of linear programming cannot be applied anymore.

representing the tradeoff between value and energy. The first objective is to minimize energy cost plus the *lost value* — which is the total value of uncompleted jobs. The second objective is to maximize the total value of completed jobs minus the energy cost.

1. For the objective of minimizing energy plus the lost value we derive a primal-dual algorithm for the single machine setting. The competitive ratio is characterized by a system of differential equations. For a specific case where $P(z) = z^\alpha$, the competitive ratio turns out to be α^α (and recognize the result in [18]). With the primal-dual framework, the result is more general and the analysis is simpler.
2. For the objective of maximizing the total value of completed jobs minus the energy cost, it has been shown that without resource augmentation no algorithm has bounded competitive ratio even for a single machine [19]. We study the problem for unrelated machines in the resource augmentation model. We give a primal-dual algorithm which is $(1 + \epsilon)$ -speed and $1/\epsilon$ -competitive for every $\epsilon \geq \epsilon(P) > 0$ where $\epsilon(P)$ depends on function P . For typical function $P(z) = z^\alpha$, $\epsilon(P) = 1 - \alpha^{-1/\alpha}$ which is closed to 0 for large α .

Note that for these problems, we consider relaxations with convex objectives and linear constraints.

Dual-fitting approach. An essential point of the primal-dual approach is a *convex* relaxation of the corresponding problems. However, some problems unlikely admit such a relaxation. To overcome that difficulty, we follow the dual-fitting approach for *non-convex* programming presented in [20]. A summary of the approach is as follows.

Given a problem, formulate a relaxation which is *not* necessarily convex and its Lagrangian dual. Next construct dual variables such that the Lagrangian dual has objective value within a desired factor of the primal one (due to some algorithm). Then by the standard Lagrangian weak duality² for mathematical programming, the competitive ratio follows. Since the Lagrangian weak duality also holds in the context of calculus of variations, the approach could be applied for the unknowns which are not only variables but also functions.

Let $L(x, \lambda)$ be the Lagrangian function with primal and dual variables x and λ , respectively. Let \mathcal{X} and \mathcal{Y} are feasible sets of x and λ . Intuitively, the approach could be seen as a game between an algorithm and an adversary. The algorithm chooses dual variables λ^* in such a way that whatever the choice (strategy) of the adversary, the value $\min_{x \in \mathcal{X}} L(x, \lambda^*)$ is always within a desirable factor c of the objective due to the algorithm. We emphasize that $\min_{x \in \mathcal{X}} L(x, \lambda^*)$ is taken over x feasible solutions of the primal.

An advantage of the approach is the flexibility of the formulation. As convexity is not required, we can come up with a direct and natural relaxation for the problem. The main core of the approach is to determine the dual variables and to prove the desired competitive ratio. Determining such dual variables is the crucial step in the analysis. However, the dual variables usually have intuitive interpretations which are useful to figure out appropriate values of such variables. Besides, the dual variables are not interactively constructed as in the primal-dual approach — this is the main difference between two approaches. Nevertheless, for some problems one could informally separate the convex and non-convex parts. Then the dual solution for the original problem may be derived

²For completeness, the proof of weak duality is given in the appendix

from a dual solution for the convex part (constructed using primal-dual) by adding some correcting terms due to the non-convex part.

Applications. We consider the general energy model: speed scaling with power down. There is a machine which can be set either in the sleep state or in the active state. Each transition of the machine from the sleep state to the active one has cost A , which represents the *wake-up* cost. In the sleep state, the energy consumption of the machine is 0. The machine, in its active state, can choose a speed $s(t)$ to execute jobs. The power energy consumption of the machine at time t in its active state is $P(s(t)) = s(t)^\alpha + g$ where $\alpha \geq 1$ and $g \geq 0$ are characteristic parameters of the machine. Hence, the consumed energy (without wake-up cost) of the machine is $\int_0^\infty P(s(t))dt$ where the integral is taken during the machine's active periods. We decompose the latter into *dynamic energy* $\int_0^\infty s^\alpha(t)dt$ and *static energy* $\int_0^\infty gdt$ (where again the integrals are taken during active periods). Jobs arrive over time, a job j is released at time r_j , has weight w_j and requires p_j units of processing volume if it is processed on machine i . A job could be processed preemptively. At any time, the scheduler has to determine the state and the speed of every machine (if it is active) and also a policy to execute jobs. We consider two problems in the setting.

In the first problem, each job j has additionally a deadline d_j by which the job has to be completed. The objective is to minimize the total consumed energy.

In the second problem, jobs do not have deadline. Let C_j be the completion time of the job j . The *flow-time* of a job j is defined as $C_j - r_j$, which represented the waiting time of j on the server. The objective is to minimize the total weighted flow-time of all jobs plus the total energy.

As posed in [1], an important direction in energy-efficient scheduling is to design competitive algorithms for online problems in the general model of speed scaling with power down. Attempting efficient algorithms in the general energy model, one encounters the limits of current tools which have been successfully applied in previous energy models. That results in a few work on the model [2, 4, 13], in contrast to the widely-studied models of speed scaling only or power down only. The potential function method, as mentioned earlier, yield little insight on the construction of new algorithms in this general setting. Besides, different proposed approaches based on the duality of mathematical programming [3, 12, 10] require that the problems admit linear or convex relaxations. However, it is unlikely to formulate problems in the general energy model as convex programs.

Our results in the general energy model are the following.

1. For the problem of minimizing the total consumed energy, we formulate a natural *non-convex* relaxation using the Dirac delta function. We first revisit a special case with no wake-up cost under the primal-dual view. In this case, the relaxation becomes convex and our framework could be applied to show a α^α -competitive algorithm (the algorithm is in fact algorithm OPTIMAL AVAILABLE [21]). Next we study the general problem with wake-up cost. The special case effectively gives ideas to determine the machine speed in active state. Thus we consider an algorithm in which the procedure maintaining the machine speed in active state follows the ideas in the special case. The algorithm turns out to be algorithm SLEEP-AWARE OPTIMAL AVAILABLE (SOA) [13] with different description (due to the primal-dual view). Han et al. [13] proved that SOA has competitive ratio $\max\{4, \alpha^\alpha + 2\}$. We prove that SOA is indeed $\max\{4, \alpha^\alpha\}$ -competitive by the dual-fitting technique. Although the improvement is slight, the analysis is *tight*³ and it suggests that the duality-based approach is seemingly a

³The algorithm has competitive ratio exactly α^α even without wake-up cost [5].

right tool for online scheduling. Through the problem, we illustrate an interesting feature in the construction of algorithms for non-convex relaxations. The primal-dual framework gives ideas for the design of an algorithm while the analysis is done using dual-fitting technique.

2. For the problem of minimizing energy plus weighted flow-time, we derive a $O(\alpha/\log \alpha)$ -competitive algorithm using the dual fitting framework; that matches the best known competitive ratio (up to a constant) for the same problem in the restricted speed scaling model (where the wake-up cost and the static energy cost are 0). Informally, the dual solutions are constructed as the combination of a solution for the convex part of the problem and a term that represents the lost due to the non-convex part. Building upon the salient ideas of the previous analysis, we manage to show the competitiveness of the algorithm.

1.2 Related work

In the search for principled methods to design and analyze online problems, especially in online scheduling, interesting approaches [3, 12, 20] based on mathematical programming have been presented. The approaches give insight about the nature of many scheduling problems, hence lead to algorithms which are usually simple and competitive [3, 12, 20, 10, 15, 14].

Anand et al. [3] was the first who proposed studying online scheduling by linear (convex) programming and dual fitting. By this approach, they gave simple algorithms and simple analyses with improved performance for problems where the analyses based on potential functions are complex or it is unclear how to design such functions. Subsequently, Nguyen [20] generalized the approach in [3] and proposed to study online scheduling by non-convex programming and the weak Lagrangian duality. Using that technique, [20] derive competitive algorithms for problems related to weighted flow-time.

Buchbinder and Naor [8] presented the primal-dual method for online packing and covering problems. Their method unifies several previous potential function based analyses and is a powerful tool to design and analyze algorithms for problems with linear relaxations. Gupta et al. [12] gave a primal-dual algorithm for a general class of scheduling problems with cost function $f(z) = z^\alpha$. Devanur and Jain [11] also used the primal-dual approach to derive optimal competitive ratios for online matching with concave return. The construction of dual programs in [10, 11] is based on convex conjugates and Fenchel duality for primal convex programs in which the objective is convex and the constraints are *linear*.

An interesting quality of service in online scheduling is the tradeoff between energy and throughput. The online problem to minimize the consumed energy plus lost values with the energy power $P(z) = z^\alpha$ is first studied by [9] where a $(\alpha^\alpha + 2e\alpha)$ -competitive algorithm is given for a single machine. Subsequently, Kling and Pietrzyk [18] derived an improved α^α -competitive for identical machines with migration using the technique in [12]. The online problem to maximize the total value of completed jobs minus the consumed energy for a single machine has been considered in [19]. Pruhs and Stein [19] proved that the competitive ratio without resource augmentation is unbounded and gave an $(1 + \epsilon)$ -speed, $O(1/\epsilon^3)$ -competitive algorithm for a single machine.

The objective of minimizing the total flow-time plus energy has been widely studied in speed scaling energy model. For a single machine, Bansal et al. [6] gave a $(3 + \epsilon)$ -competitive algorithm. Besides, they also proved a $(2 + \epsilon)$ -competitive algorithm for minimizing total *fractional* weighted flow-time plus energy. Their results hold for a general class of convex power functions. Those results also imply an $O(\alpha/\log \alpha)$ -competitive algorithm for weighted flow-time plus energy when

the energy function is s^α . Again, always based on linear programming and dual-fitting, Anand et al. [3] proved an $O(\alpha^2)$ -competitive algorithm for unrelated machines. Subsequently, Nguyen [20] and Devanur and Huang [10] presented an $O(\alpha/\log \alpha)$ -competitive algorithms for unrelated machines by dual fitting and primal dual approaches, respectively. It turns out that the different approaches lead to the same algorithm. To the best of our knowledge, this objective is not studied in the speed scaling with power down energy model.

In the speed scaling with power down energy model, all previous papers considered the problem of minimizing the energy consumption on a single machine. Irani et al. [16] was the first who studied the problem in online setting and derived an algorithm with competitive ratio $(2^{2\alpha-2}\alpha^\alpha + 2^{\alpha-1} + 2)$. Subsequently, Han et al. [13] presented an algorithm which is $\max\{4, \alpha^\alpha + 2\}$ -competitive. In offline setting, the problem is recently showed to be NP-hard [2]. Moreover, Albers and Antoniadis [2] also gave a 1.171-approximation algorithm, which improved the 2-approximation algorithm in [16]. If the instances are agreeable then the problem is polynomial [4].

1.3 Organization

The paper is organized as follows. In Section 2, we introduce the online convex assignment problem and present a primal-dual framework for this problem. In Section 3 and Section 4, we apply the framework to derive primal-dual algorithms for problems related to the tradeoff between energy and value. In Section 5 and Section 6, we study problems in the speed scaling with power down model using the dual-fitting approach. In the former, we study the problem of minimizing energy and in the latter we consider the problem of minimizing the total energy plus weighted flow-time. In the beginning of each section, we restate the considered problem in a short description.

2 Framework for Online Convex Assignment

Consider the assignment problem where items j arrive online and need to be (fractionally) assigned to some agents i with the following objective and constraints.

$$\begin{aligned}
\min \quad & P(x) := \sum_i f_i \left(\sum_j a_{ij} x_{ij} \right) \\
\text{subject to} \quad & g_i \left(\sum_j b_{ij} x_{ij} \right) \leq 0 \quad \forall i \\
& h_j \left(\sum_i c_{ij} x_{ij} \right) \leq 0 \quad \forall j \\
& x_{ij} \geq 0 \quad \forall i, j
\end{aligned} \tag{P}$$

where x_{ij} indicates the amount of item j assigned to agent i and functions f_i, g_i, h_j are convex, differential for every i, j and $a_{ij}, b_{ij} \geq 0$. Denote $k \prec j$ if item k is released before item j .

Let \mathcal{X} be the set of feasible solutions of (P). The Lagrangian dual is $\max_{\lambda, \gamma \geq 0} \min_{x \in \mathcal{F}} L(x, \lambda, \gamma)$

where L is the following Lagrangian function

$$\begin{aligned}
L(x, \lambda, \gamma) &= \sum_i f_i \left(\sum_j a_{ij} x_{ij} \right) + \sum_i \lambda_i g_i \left(\sum_j b_{ij} x_{ij} \right) + \sum_j \gamma_j h_j \left(\sum_i c_{ij} x_{ij} \right) \\
&\geq \sum_{i,j} (x_{ij} - x_{ij}^*) \left[a_{ij} f'_i \left(\sum_k a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_k b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_{i'} c_{i'j} x_{i'j}^* \right) \right] \\
&\quad + \sum_i f_i \left(\sum_j a_{ij} x_{ij}^* \right) + \sum_i \lambda_i g_i \left(\sum_j b_{ij} x_{ij}^* \right) + \sum_j \gamma_j h_j \left(\sum_i c_{ij} x_{ij}^* \right) \\
&\geq \sum_{i,j} (x_{ij} - x_{ij}^*) \left[a_{ij} f'_i \left(\sum_{k \prec j} a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_{k \prec j} b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_{i'} c_{i'j} x_{i'j}^* \right) \right] \\
&\quad + \sum_i f_i \left(\sum_j a_{ij} x_{ij}^* \right) + \sum_i \lambda_i g_i \left(\sum_j b_{ij} x_{ij}^* \right) + \sum_j \gamma_j h_j \left(\sum_i c_{ij} x_{ij}^* \right)
\end{aligned}$$

where the inequalities holds for any x^* due the convexity of functions f_i, g_i, h_j 's. In the first inequality, we use $f_i(z) \geq f_i(z^*) + (z - z^*) f'_i(z^*)$ (similarly for functions g_i, h_j 's) and in the second inequality, we use the monotonicity of f'_i (and similarly for g'_i). Denote

$$\begin{aligned}
M(x, x^*, \lambda, \gamma) &:= \sum_{i,j} (x_{ij} - x_{ij}^*) \left[a_{ij} f'_i \left(\sum_{k \prec j} a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_{k \prec j} b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_i c_{ij} x_{ij}^* \right) \right] \\
N(x^*, \lambda, \gamma) &:= \sum_i f_i \left(\sum_j a_{ij} x_{ij}^* \right) + \sum_i \lambda_i g_i \left(\sum_j b_{ij} x_{ij}^* \right) + \sum_j \gamma_j h_j \left(\sum_i c_{ij} x_{ij}^* \right)
\end{aligned}$$

We have

$$L(x, \lambda, \gamma) \geq M(x, x^*, \lambda, \gamma) + N(x^*, \lambda, \gamma) \quad (1)$$

Intuitively, one could imagine that x^* is the solution of an algorithm (or a function on the solution of an algorithm). We emphasize that x^* is *not* a solution of an optimal assignment. The goal is to design an algorithm, which produces x^* and derives dual variables λ, γ , in such a way that the primal objective is bounded by a desired factor from the dual one.

Inequality (1) naturally leads to the following idea of an algorithm. For any item j , we maintain the following invariants

$$\begin{aligned}
a_{ij} f'_i \left(\sum_{k \prec j} a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_{k \prec j} b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_i c_{ij} x_{ij}^* \right) &\geq 0 \quad \forall i \\
a_{ij} f'_i \left(\sum_{k \prec j} a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_{k \prec j} b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_i c_{ij} x_{ij}^* \right) &= 0 \text{ if } x_{ij}^* > 0
\end{aligned}$$

Whenever the invariants hold for every j , $M(x, x^*, \lambda, \gamma) \geq 0$ since $x_{ij} \geq 0$ for every i, j . Therefore, $L(x, \lambda, \gamma) \geq N(x^*, \lambda, \gamma)$ and so the dual is lower-bounded by $N(x^*, \lambda, \gamma)$, which does not depend anymore on x . The procedure of maintaining the invariants dictate the decision x^* of an algorithm and indicates the choice of dual variables.

Consider the following dual

$$\begin{aligned}
& \max \quad N(x^*, \lambda, \gamma) \tag{D} \\
\text{subject to} \quad & a_{ij} f'_i \left(\sum_{k \prec j} a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_{k \prec j} b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_i c_{ij} x_{ij}^* \right) \geq 0 \quad \forall i, j \\
& a_{ij} f'_i \left(\sum_{k \prec j} a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left(\sum_{k \prec j} b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left(\sum_i c_{ij} x_{ij}^* \right) = 0 \text{ if } x_{ij}^* > 0 \quad \forall i, j \\
& x^*, \lambda, \gamma \geq 0 \quad \forall i, j
\end{aligned}$$

Lemma 1 (Weak Duality) *Let $OPT(\mathcal{P})$ and $OPT(\mathcal{D})$ be optimal values of primal program (\mathcal{P}) and dual program (\mathcal{D}) , respectively. Then $OPT(\mathcal{P}) \geq OPT(\mathcal{D})$.*

Proof It holds that

$$OPT(\mathcal{P}) \geq \max_{\lambda, \gamma \geq 0} \min_{x \in \mathcal{X}} L(x, \lambda, \gamma) \geq N(x^*, \lambda, \gamma)$$

where the inequalities follow the weak Lagrangian duality and the constraints of (\mathcal{D}) for every feasible solution x^*, λ, γ . Therefore, the lemma follows. \square

Hence, our framework consists of maintaining the invariants for every online item j and among feasible set of dual variables (constrained by the invariants) choose the ones which optimize the ratio between the primal and dual values. If an algorithm with output x^* satisfies $P(x^*) \leq rN(x^*, \lambda, \gamma)$ for some factor r then the algorithm is r -competitive.

3 Minimizing Total Energy plus Lost Values

The problem. We are given a machine with a convex energy power P and jobs arrive over time. Each job j is released at time r_j , has deadline d_j , processing volume p_j and a value a_j . Jobs could be executed preemptively and at any time t , the scheduler has to choose a set of *pending* jobs (i.e., $r_j \leq t < d_j$) and a machine speed $s(t)$ in order to process such jobs. The *energy cost* of a schedule is $\int_0^\infty P(s(t))dt$. Typically, $P(z) = z^\alpha$ for some constant $\alpha \geq 1$. The objective of the problem is to minimize energy cost plus the *lost value* — which is the total value of uncompleted jobs.

Formulation. Let x_j and y_j be variables indicating whether job j is completed or it is not. We denote variable $s_j(t)$ as the speed that the machine processes job j at time t . The problem could be relaxed as the following convex program.

$$\begin{aligned}
& \min \quad \int_0^\infty P(s(t))dt + \sum_j a_j y_j \\
\text{subject to} \quad & s(t) = \sum_j s_j(t) \quad \forall t \\
& x_j + y_j \geq 1 \quad \forall j \\
& \int_{r_j}^{d_j} s_j(t)dt \geq p_j x_j \quad \forall j \\
& x_j, y_j, s_j(t) \geq 0 \quad \forall j, t
\end{aligned}$$

In the relaxation, the second constraint indicates that either job j is completed or it is not. The third constraint guarantees the necessary amount of work done in order to complete job j .

Applying the framework, we have the following dual.

$$\max \int_0^\infty P\left(\sum_j v_j^*(t)\right) dt - \sum_j \lambda_j \int_{r_j}^{d_j} v_j^*(t) dt + \sum_j \gamma_j$$

subject to

1. For any job j , $\gamma_j \leq p_j \lambda_j$. Moreover, if $x_j^* > 0$ then $\gamma_j = p_j \lambda_j$.
2. For any job j , $\gamma_j \leq a_j$ and if $y_j^* > 0$ then $\gamma_j = a_j$.
3. For any job j and any $t \in [r_j, d_j]$, it holds that $\lambda_j \leq P'(\sum_k v_k^*(t))$. Particularly, if $v_j^*(t) > 0$ then $\lambda_j = P'(\sum_k v_k^*(t))$.

Note that $v_j^*(t)$ is not equal to $s_j^*(t)$ (the machine speed on job j according to our algorithm) but it is a function depending on $s_j^*(t)$. That is the reason we use $v_j^*(t)$ instead of $s_j^*(t)$. We will choose $v_j^*(t)$'s in order to optimize the competitive ratio. To simplify the notation, we drop out the star symbol in the superscript of every variable (if one has that).

Algorithm. The dual constraints naturally leads to the following algorithm. We first describe informally the algorithm. In the algorithm, we maintain a variable $u_j(t)$ representing the *virtual* machine speed on job j . The virtual speed on job j means that job j will be processed with that speed *if* it is accepted; otherwise, the real speed on j will be set to 0. Consider the arrival of job j . Observe that by the third dual constraint, we should always increase the machine speed on job j at $\arg \min P'(v(t))$ in order to increase λ_j . Hence, at the arrival of a job j , increase continuously the *virtual* speed $u_j(t)$ of job j at $\arg \min P'(v(t))$ for $r_j \leq t \leq d_j$. Moreover, function $v(t)$ is also simultaneously updated as a function of $u(t) = \sum_{k \leq j} u_k(t)$ according to a system of differential equations (2) in order to optimize the competitive ratio. The iteration on job j terminates whether one of the first two constraints becomes tight. If the first one holds, then accept the job and set the real speed equal to the virtual one. Otherwise, reject the job.

Define $Q(z) := P(z) - zP'(z)$. Consider the following system of differential equations with boundary conditions: $Q(v) = 0$ if $u = 0$.

$$\begin{aligned} Q'(v) \frac{dv}{du} + P'(v) &\geq \frac{P'(u)}{r}, \\ (r-1)P'(u) + rQ'(v) \frac{dv}{du} &\geq 0, \\ \frac{dv}{du} &> 0, \end{aligned} \tag{2}$$

where r is some constant. Let $r^* \geq 1$ be a smallest constant such that the system has a solution.

The formal algorithm is given in Algorithm 1.

In the algorithm, machine i processes accepted job j with speed $s_j(t)$ at time t . As the algorithm completes all accepted jobs, it is equivalent to state that the machine processes accepted jobs in the earliest deadline first fashion with speed $s(t)$ at time t .

By the algorithm, the dual variables are feasible. In the following we bound the values of the primal and dual objectives.

Algorithm 1 Minimizing the consumed energy plus lost values.

```
1: Initially, set  $s(t), s_j(t), u_j(t), v(t)$  and  $v_j(t)$  equal to 0 for every  $j$ .
2: Let  $r^* \geq 1$  be the smallest constant such that (2) has a solution. During the algorithm, keep
    $v(t)$  as a solution of (2) with constant  $r^*$  and  $u(t) = \sum_j u_j(t)$  for every time  $t$ .
3: for a job  $j$  arrives do
4:   Initially,  $u_j(t) \leftarrow 0$ .
5:   while  $\int_{r_j}^{d_j} u_j(t)dt < p_j$  and  $\lambda_j p_j < a_j$  do
6:     Continuously increase  $u_j(t)$  at  $\arg \min P'(v(t))$  for  $r_j \leq t \leq d_j$  and update  $u(t) \leftarrow$ 
        $\sum_{k \neq j} u_k(t) + u_j(t)$  and  $v(t)$  (as a function of  $u(t)$ ) and  $\lambda_j \leftarrow \min_{r_j \leq t \leq d_j} P'(v(t))$  simulta-
       neously.
7:   end while
8:   Set  $v_j(t) \leftarrow v(t) - \sum_{k \prec j} v_k(t)$ .
9:   if  $\lambda_j p_j = a_j$  and  $\int_{r_j}^{d_j} u_j(t)dt < p_j$  then
10:    Reject job  $j$ .
11:    Set  $\gamma_j \leftarrow a_j$ .
12:   else
13:    Accept job  $j$ .
14:    Set  $s_j(t) \leftarrow u_j(t)$ ,  $s(t) \leftarrow s(t) + s_j(t)$  and  $\gamma_j \leftarrow \lambda_j p_j$ .
15:   end if
16: end for
```

Lemma 2 *It holds that*

$$\int_0^\infty P(s(t))dt + \sum_j a_j y_j \leq r^* \left(\int_0^\infty P\left(\sum_j v_j(t)\right)dt - \sum_j \lambda_j \int_{r_j}^{d_j} v_j(t)dt + \sum_j \gamma_j \right)$$

Proof By the algorithm, $\lambda_j = P'(v(t))$ at every time t such that $v_j(t) > 0$ for every job j . Hence, it is sufficient to show that

$$\int_0^\infty P(s(t))dt + \sum_j a_j y_j \leq r^* \left(\int_0^\infty Q\left(\sum_j v_j(t)\right)dt + \sum_j \gamma_j \right) \quad (3)$$

where recall $Q(z) = P(z) - zP'(z)$.

We will prove the inequality (3) by induction on the number of jobs in the instance. For the base case where there is no job, the inequality holds trivially. Suppose that the inequality holds before the arrival of a job j . In the following, we consider different cases.

Job j is accepted. Consider any moment τ in the while loop related to job j . We emphasize that τ is a moment in the execution of the algorithm, not the one in the time axis t . Suppose that at moment τ , an amount $du_j(t)$ is increased (allocated) at t . Note that $du_j(t) = du(t)$ as $u(t) = \sum_j u_j(t)$. As j is accepted, $y_j = 0$ and the increase at τ in the left hand-side of (3) is $P'(u(t))du(t)$

Let $v(t_1, \tau_1)$ be the value of $v(t_1)$ at moment τ_1 in the while loop. By the algorithm, the dual variable γ_j satisfies

$$\begin{aligned}\gamma_j &= \lambda_j p_j \geq \int_{\tau_1} \left(\int_{r_j}^{d_j} u_j(t_2) dt_2 \right) \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau_1)) d\tau_1 \\ &= \int_{\tau_1} \int_{r_j}^{d_j} u_j(t_2) \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau_1)) dt_2 d\tau_1\end{aligned}$$

where the inequality is due to the fact that at the end of the while loop, $\int_{r_j}^{d_j} u_j(t) dt = p_j$ (by the loop condition) and P' is increasing. Therefore, at moment τ , $d\gamma_j \geq \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau)) du_j(t) = P'(v(t)) du(t)$ where the equality follows since $t \in \arg \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau))$. Hence, the increase in the right hand-side of (3) is at least $r^*[Q'(v(t))dv(t) + P'(v(t))du(t)]$.

Due to the system of inequations (2) and the choice of r^* , at any moment in the execution of the algorithm, the increase in the left hand-side of (3) is at most that in the right hand-side. Thus, the induction step follows.

Job j is rejected. If j is rejected then $y_j = 1$ and so the increase in the left hand-side of (3) is a_j . Moreover, by the algorithm $\gamma_j = a_j$. So we need to prove that after the iteration of the for loop on job j , it holds that $(r^* - 1)a_j + r^* \int_0^\infty \Delta Q(v(t)) dt \geq 0$. As j is rejected,

$$a_j = \lambda_j p_j > \int_{r_j}^{d_j} \min_{r_j \leq t_1 \leq d_j} P'(v(t_1)) u_j(t) dt$$

Therefore, it is sufficient to prove that

$$(r^* - 1) \int_{r_j}^{d_j} \min_{r_j \leq t_1 \leq d_j} P'(v(t_1)) u_j(t) dt + r^* \int_0^\infty \Delta Q(v(t)) dt \geq 0 \quad (4)$$

Before the iteration of the while loop, the left-hand side of (4) is 0. Similar as the analysis of the previous case, during the execution of the algorithm the increase rate of the left-hand side is $(r^* - 1)P'(v(t))du(t) + r^*Q'(v(t))dv(t)$, which is non-negative by equation (2). Thus, inequality (4) holds.

By both cases, the lemma follows. \square

Theorem 1 *The algorithm is r^* -competitive. Particularly, if the energy power function $P(z) = z^\alpha$ then the algorithm is α^α -competitive*

Proof The theorem follows by the framework and Lemma 2.

If the power energy function $P(z) = z^\alpha$ then $r^* = \alpha^\alpha$ and $u(t) = v(t)/\alpha$ satisfy the system (2). Thus, the algorithm is α^α -competitive. \square

4 Maximizing the Total Value minus Energy

The problem. We are given unrelated machines and jobs arrive over time. Each job j is released at time r_j , has deadline d_j , a value a_j and processing volume p_{ij} if it is executed on machine i .

Jobs could be executed preemptively but migration is not allowed, i.e., no job could be executed in more than one machine. At a time t , the scheduler has to choose a set of *pending* jobs (i.e., $r_j \leq t < d_j$) to be processed on each machine, and the speeds $s_i(t)$'s for every machine i to execute such jobs. The energy cost is $\sum_i \int_0^\infty P(s_i(t))dt$ where P is a given convex power function. The objective now is to maximize the total value of completed jobs minus the energy cost.

We first give some idea about the difficulty of the problem even on a single machine. Assume that the adversary releases a job with small value but with a high energy cost in order to complete the job. One has to execute the job since otherwise the profit would be zero. However, at the moment an algorithm nearly completes the job, the adversary releases other job with much higher value and a reasonable energy demand. One need to switch immediately to the second job since otherwise either a high value is lost or the energy consumption becomes too much. It means that all energy spending on the first job is lost without any gain. Based on this idea, [19] has shown that without resource augmentation, the competitive ratio is unbounded.

In this section, we consider the problem with resource augmentation, meaning that with the same speed z the energy power for the algorithm is $P((1 - \epsilon)z)$, whereas the one for the adversary is $P(z)$. Let $\epsilon(P) > 0$ be the smallest constant such that $zP'((1 - \epsilon(P))z) \leq P(z)$ for all $z > 0$. For the typical energy power $P(z) = z^\alpha$, $\epsilon(P) = 1 - \alpha^{-1/\alpha}$ which is closed to 0 for α large.

Formulation. Let x_{ij} be variable indicating whether job j is completed in machine i . Let $s_{ij}(t)$ be the variable representing the speed that machine i processes job j at time t . The problem could be formulated as the following convex program.

$$\begin{aligned}
& \max \quad \sum_{i,j} a_j x_{ij} - \sum_i \int_0^\infty P((1 - \epsilon)s_i(t))dt \\
& \text{subject to} \quad s_i(t) = \sum_j s_{ij}(t) & \forall i, t \\
& \quad \sum_i x_{ij} \leq 1 & \forall j \\
& \quad \int_{r_j}^{d_j} s_{ij}(t)dt \geq p_{ij} x_{ij} & \forall i, j \\
& \quad x_{ij}, s_{ij}(t) \geq 0 & \forall i, j, t
\end{aligned}$$

Note that in the objective, by resource augmentation the consumed energy is $\sum_i \int_0^\infty P((1 - \epsilon)s_i(t))dt$. Applying the framework, we have the following dual.

$$\min \sum_j \gamma_j - \sum_i \int_0^\infty P\left(\sum_j u_{ij}^*(t)\right)dt + \sum_{i,j} \lambda_{ij} \int_{r_j}^{d_j} u_{ij}^*(t)dt$$

subject to

1. For any machine i and any job j , $\gamma_j + p_{ij}\lambda_{ij} \geq a_j$.
2. For any machine i , any job j and any $t \in [r_j, d_j]$, $\lambda_{ij} \leq P'((1 - \epsilon)\sum_k u_{ik}^*(t))$ where the sum is taken over all jobs k released before j , i.e., $k \preceq j$. Particularly, if $u_{ij}^*(t) > 0$ then $\lambda_{ij} = P'((1 - \epsilon)\sum_{k \preceq j} u_{ik}^*(t))$.

Similar as in the previous section, the constraints naturally lead to Algorithm 2. In the algorithm and the analysis, to simplify the notation we drop out the star symbol in the superscript of every variable (if one has that).

Algorithm 2 Minimizing the throughput minus consumed energy.

```

1: Initially, set  $s(t)$  and  $u(t)$  equal to 0.
2: The algorithm always runs accepted jobs with speed  $s(t)$  in the earliest deadline first fashion.
3: for a job  $j$  arrives do
4:   Initially,  $u_{ij}(t) \leftarrow 0$  for every  $t$  and let  $\mathcal{I}$  be the set of all machines,  $\mathcal{I}' \leftarrow \emptyset$ .
5:   while  $\mathcal{I} \neq \emptyset$  do
6:     For every  $i \in \mathcal{I}$ , increase  $u_{ij}(t)$  at  $\arg \min P'(u_i(t))$  in the continuous manner for  $r_j \leq t \leq d_j$  and update  $u_i(t) = \sum_{k \neq j} u_{ik}(t) + u_{ij}(t)$  and  $\lambda_{ij} = \min_{r_j \leq t \leq d_j} P'((1 - \epsilon)u_i(t))$  simultaneously.
7:     if  $\lambda_{ij}p_{ij} = a_j$  and  $\int_{r_j}^{d_j} u_{ij}(t)dt < p_j$  for some machine  $i$  then
8:        $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$ .
9:     end if
10:    if  $\lambda_{ij}p_{ij} < a_j$  and  $\int_{r_j}^{d_j} u_{ij}(t)dt = p_j$  for some machine  $i$  then
11:       $\mathcal{I}' \leftarrow \mathcal{I}' \cup \{i\}$  and  $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$ .
12:    end if
13:  end while
14:  if  $\mathcal{I}' = \emptyset$  then
15:    Reject job  $j$  and set  $\gamma_j \leftarrow 0$  (note that  $p_{ij}\lambda_{ij} = a_j \ \forall i$ ).
16:  else
17:    Let  $i = \arg \min_{i' \in \mathcal{I}'} p_{i'j}\lambda_{i'j}$ .
18:    Accept and assign job  $j$  to machine  $i$ , i.e.,  $x_{ij} = 1$ .
19:    Set  $s_{ij}(t) \leftarrow u_{ij}(t)$ ,  $s_i(t) \leftarrow s_i(t) + s_{ij}(t)$  and  $\gamma_j \leftarrow a_j - \lambda_j p_j$ .
20:  end if
21: end for

```

Lemma 3 *Dual variables constructed by Algorithm 2 are feasible.*

Proof By the algorithm (line 6), $\lambda_{ij} \leq P'((1 - \epsilon) \sum_k u_{ik}(t))$ where the sum is taken over all jobs k released before j ($k \preceq j$) and if $u_{ij}(t) > 0$ then $\lambda_{ij} = P'((1 - \epsilon) \sum_{k \preceq j} u_{ik}(t))$. Consider the first constraint. If j is rejected then $p_{ij}\lambda_{ij} = a_j$ for every machine i . Otherwise, by the assignment (line 17), it always holds that $\gamma_j + p_{ij}\lambda_{ij} \geq a_j$ for every i, j . \square

In the following we bound the values of the primal and dual objectives in the resource augmentation model.

Lemma 4 *For every $\epsilon \geq \epsilon(P)$, it holds that*

$$\frac{1}{\epsilon} \left(\sum_{i,j} a_j x_{ij} - \sum_i \int_0^\infty P((1 - \epsilon)s_i(t))dt \right) \geq \sum_j \gamma_j - \sum_i \int_0^\infty P\left(\sum_j u_{ij}(t)\right)dt + \sum_{i,j} \lambda_{ij} \int_{r_j}^{d_j} u_{ij}(t)dt$$

Proof We have

$$\begin{aligned}
& \sum_{i,j} \lambda_{ij} \int_{r_j}^{d_j} u_{ij}(t) dt - \sum_i \int_0^\infty P(u_i(t)) dt \\
& \leq \sum_i \int_0^\infty \left(\max_{j:r_j \leq t \leq d_j} \lambda_{ij} \right) \sum_j u_{ij}(t) dt - \sum_i \int_0^\infty P(u_i(t)) dt \\
& \leq \sum_i \int_0^\infty P'((1-\epsilon)u_i(t)) u_i(t) dt - \sum_i \int_0^\infty P(u_i(t)) dt \leq 0
\end{aligned}$$

In the second inequality, recall that $\sum_j u_{ij}(t) = u_i(t)$ and by the algorithm, $\lambda_{ij} = \min_{r_j \leq t \leq d_j} P'((1-\epsilon)u_i(t)) \leq P'((1-\epsilon)u_i(t))$ for any $r_j \leq t \leq d_j$. The last inequality follows by the definition of $\epsilon(P)$. Therefore, it is sufficient to prove that

$$\frac{1}{\epsilon} \left(\sum_{i,j} a_j x_{ij} - \sum_i \int_0^\infty P((1-\epsilon)s_i(t)) dt \right) \geq \sum_j \gamma_j \quad (5)$$

We prove inequality (5) by induction on the number of released jobs in the instance. For the base case where there is no job, the inequality holds trivially. Suppose that the inequality holds before the arrival of a job j .

If j is rejected then $x_{ij} = 0$, $s_{ij}(t) = 0$ for every i, t and $\gamma_j = 0$. Therefore, the increases in both side of inequality (5) are 0. Hence, the induction step follows.

In the rest, assume that j is accepted and let i be the machine to which j is assigned. We have

$$\begin{aligned}
& \int_0^\infty \left[P((1-\epsilon)u_i(t)) - P((1-\epsilon)(u_i(t) - u_{ij}(t))) \right] dt \leq (1-\epsilon) \int_0^\infty P'((1-\epsilon)u_i(t)) u_{ij}(t) dt \\
& \leq (1-\epsilon) \int_0^\infty P'(u_i(t)) u_{ij}(t) dt = (1-\epsilon) \lambda_{ij} \int_0^\infty u_{ij}(t) dt = (1-\epsilon) \lambda_{ij} p_{ij} \leq (1-\epsilon) a_j
\end{aligned}$$

The first inequality is due to the convexity of P

$$P((1-\epsilon)(u_i(t) - u_{ij}(t))) \geq P((1-\epsilon)u_i(t)) - (1-\epsilon)u_{ij}(t)P'((1-\epsilon)u_i(t)).$$

The second inequality holds because P' is increasing. The first equality follows since $u_{ij}(t) \neq 0$ only at t such that $P'(u_i(t)) = \lambda_{ij}$ (by the algorithm). The last inequality is due to the loop condition in the algorithm. Thus, at the end of the iteration (related to job j) in the for loop, the increase in the left hand side of inequality (5) is

$$\frac{1}{\epsilon} \left(a_j x_{ij} - \int_0^\infty \left[P((1-\epsilon)u_i(t)) - P((1-\epsilon)(u_i(t) - u_{ij}(t))) \right] dt \right) \geq \frac{1}{\epsilon} (a_j - (1-\epsilon)a_j) = a_j$$

Besides, the increase in the right hand-side of inequality (5) is $\gamma_j \leq a_j$. Hence, the induction step follows; so does the lemma. \square

Theorem 2 *The algorithm is $(1+\epsilon)$ -augmentation, $1/\epsilon$ -competitive for $\epsilon \geq \epsilon(P)$.*

Proof By resource augmentation, with the same speed z the energy power for the algorithm is $P((1-\epsilon)z)$, whereas the one for the adversary is $P(z)$. So by Lemma 4, the theorem follows. \square

Note that the result could be generalized for *heterogeneous* machines where the energy power functions are different. In this case, one needs to consider $\epsilon \geq \max_i \epsilon(P_i)$.

5 Energy Minimization in Speed Scaling with Power Down Model

The problem. We are given a single machine that could be transitioned into a sleep state or an active state. Each transition from the sleep state to the active state costs $A > 0$, which is called the *wake-up cost*. Jobs arrive online, each job has a released time r_j , a deadline d_j , a processing volume p_j and could be processed preemptively. In the problem, all jobs have to be completed. In the sleep state, the energy consumption of the machine is 0. In the active state, the power energy consumption at time t is $P(s(t)) = s(t)^\alpha + g$ where $\alpha \geq 1$ and $g \geq 0$ are constant. Thus, the consumed energy of the machine in active state is $\int_0^\infty P(s(t))dt$, that can be decomposed into *dynamic energy* $\int_0^\infty s(t)^\alpha dt$ and *static energy* $\int_0^\infty g dt$ (where the integral is taken over t at which the machine is in active state). At any time t , the scheduler has to decide the state of the machine and the speed if the machine is in active state in order to execute and complete all jobs. The objective is to minimize the total energy — the consumed energy in active state plus the wake-up energy.

Formulation. In a mathematical program for the problem, we need to incorporate an information about the machine states and the transition cost from the sleep state to the active one. Here we make use of the properties of the Heaviside step function and the Dirac delta function to encode the machine states and the transition cost. Recall that the Heaviside step function $H(t) = 0$ if $t < 0$ and $H(t) = 1$ if $t \geq 0$. Then $H(t)$ is the integral of the Dirac delta function δ (i.e., $H' = \delta$) and it holds that $\int_{-\infty}^{+\infty} \delta(t)dt = 1$. Now let $F(t)$ be a function indicating whether the machine is in active state at time t , i.e., $F(t) = 1$ if the machine is active at t and $F(t) = 0$ if it is in the sleep state. Assume that the machine initially is in the sleep state. Then $A \int_0^{+\infty} |F'(t)|dt$ equals twice the transition cost of the machine (a transition from the active state to the sleep state costs 0 while in $A \int_0^{+\infty} |F'(t)|dt$, it costs A).

Let $s_j(t)$ be variable representing the machine speed on job j at time t . The problem could be formulated as the following (non-convex) program.

$$\begin{aligned} \min \quad & \int_0^\infty P\left(\sum_j s_j(t)\right) F(t) dt + \frac{A}{2} \int_0^{+\infty} |F'(t)| dt \\ \text{subject to} \quad & \int_{r_j}^{d_j} s_j(t) F(t) dt \geq p_j \quad \forall j \\ & s_j(t) \geq 0, F(t) \in \{0, 1\} \quad \forall j, t \end{aligned}$$

The first constraint ensures that every job j will be fully processed during $[r_j, d_j]$. Moreover, each time a job is executed, the machine has to be in the active state. Note that we do not relax the variable $F(t)$. The objective function consists of the energy cost during the active periods and the wake-up cost.

5.1 Speed Scaling without Wake-Up Cost

The problem without wake-up cost ($A = 0$) has been extensively studied. We reconsider the problem throughout our primal-dual approach. In case $A = 0$, the machine is put in active state whenever there is some pending job (thus the function $F(t)$ is useless and could be removed from

the formulation). In this case, the relaxation above becomes a convex program. Applying the framework and by the same observation as in previous sections, we derive the following algorithm.

At the arrival of job j , increase continuously $s_j(t)$ at $\arg \min P'(s(t))$ for $r_j \leq t \leq d_j$ and update simultaneously $s(t) \leftarrow s(t) + s_j(t)$ until $\int_{r_j}^{d_j} s_j(t') dt' = p_j$.

It turns out that the machine speed $s(t)$ of the algorithm equals $\max_{t' > t} V(t, t') / (t' - t)$ where $V(t, t')$ is the remaining processing volume of jobs arriving at or before t with deadline in $(t, t']$. So the algorithm is indeed algorithm OPTIMAL AVAILABLE [21] that is α^α -competitive [5]. However, the primal-dual view of the algorithm gives more insight and that is useful in the general energy model (see Lemma 5).

5.2 Speed Scaling with Wake-Up Cost

The Algorithm. Define the *critical* speed $s^c = \arg \min_{s > 0} P(s)/s$. In the algorithm, the machine speed is always at least s^c if it executes some job.

Initially, set $s(t)$ and $s_j(t)$ equal 0 for every time t and jobs j . If a job is released then it is marked as *active*. Intuitively, a job is active if its speed $s_j(t)$ has not been settled yet. Let τ be the current moment. Consider currently active jobs in the earliest deadline first (EDF) order. Increase continuously $s_j(t)$ at $\arg \min P'(s(t))$ for $r_j \leq t \leq d_j$ and update simultaneously $s(t) \leftarrow s(t) + s_j(\tau)$ until $\int_{r_j}^{d_j} s_j(t') dt' = p_j$. Now consider different states of the machine at the current time τ . We distinguish three different states: (1) in *working state* the machine is active and is executing some jobs; (2) in *idle state* the machine is active but its speed equals 0; and (3) in *sleep state* the machine is inactive.

In working state. If $s(\tau) > 0$ then keep process jobs with the earliest deadline by speed $\max\{s(\tau), s^c\}$.

Mark all currently pending jobs as inactive. If $s(\tau) = 0$, switch to the idle state.

In idle state. If $s(\tau) \geq s^c$ then switch to the working state.

If $s^c > s(t) > 0$. Mark all currently pending jobs as active. Intuitively, we delay such jobs until some moment where the machine has to run at speed s^c in order to complete these jobs (assuming that there is no new job released).

Otherwise, if the total duration of idle state from the last wake-up equals A/g then switch to the sleep state.

In sleep state. If $s(t) \geq s^c$ then switch to the working state.

In the rest, we denote $s^*(t)$ as the machine speed at time t by the algorithm. Moreover, let $s_j^*(t)$ be the speed of the algorithm on job j at time t .

Analysis. The Lagrangian dual is $\max_{\lambda \geq 0} \min_{s, F} L(s, F, \lambda)$ where the minimum is taken over (s, F) feasible solutions of the primal and L is the following Lagrangian function

$$\begin{aligned} L(s, F, \lambda) &= \int_0^\infty P\left(\sum_j s_j(t)\right) F(t) dt + \frac{A}{2} \int_0^{+\infty} |F'(t)| dt + \sum_j \lambda_j \left(p_j - \int_{r_j}^{d_j} s_j(t) F(t) dt \right) \\ &\geq \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{d_j} s_j(t) F(t) \left(\lambda_j - \frac{P(s(t))}{s(t)} \right) \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) = 1\}} dt + \frac{A}{2} \int_0^{+\infty} |F'(t)| dt \end{aligned}$$

where $s(t) = \sum_j s_j(t)$.

By weak duality, the optimal value of the primal is always larger than the one of the corresponding Lagrangian dual. In the following, we bound the Lagrangian dual value in function of the algorithm cost and derive the competitive ratio via the dual-fitting approach.

Dual variables Let $0 < \beta \leq 1$ be some constant to be chosen later. For jobs j such that $s^*(t) > 0$ for every $t \in [r_j, d_j]$, define λ_j such that $\lambda_j p_j / \beta$ equals the marginal increase of the *dynamic* energy due to the arrival of job j . For jobs j such that $s^*(t) = 0$ for some moment $t \in [r_j, d_j]$, define λ_j such that $\lambda_j p_j$ equals the marginal increase of the *dynamic and static* energy due to the arrival of job j .

Lemma 5 *Let j be an arbitrary job.*

1. *If $s^*(t) > 0$ for every $t \in [r_j, d_j]$ then $\lambda_j \leq \beta P'(s^*(t))$ for every $t \in [r_j, d_j]$.*
2. *Moreover, if $s^*(t) = 0$ for some $t \in [r_j, d_j]$ then $\lambda_j = P(s^c) / s^c$.*

Proof We prove the first claim. For any time t , speed $s^*(t)$ is non-decreasing as long as new jobs arrive. Hence, it is sufficient to prove the claim assuming that no other job is released after j . So $s^*(t)$ is the machine speed after the arrival of j . The marginal increase in the dynamic energy due to the arrival of j could be written as

$$\begin{aligned} \frac{1}{\beta} \lambda_j p_j &= \int_{r_j}^{d_j} \left(P(s^*(t)) - P(s^*(t) - s_j^*(t)) \right) dt \leq \int_{r_j}^{d_j} P'(s^*(t)) s_j^*(t) dt \\ &= \min P'(s^*(t)) \int_{r_j}^{d_j} s_j^*(t) dt = \min P'(s^*(t)) p_j \end{aligned}$$

where $\min P'(s^*(t))$ is taken over $t \in [r_j, d_j]$ such that $s_j^*(t) > 0$. The inequality is due to the convexity of P and the second equality follows by the algorithm. Moreover, $\min_{r_j \leq t \leq d_j} P'(s^*(t)) \leq P'(s^*(\tau))$ for any $\tau \in [r_j, d_j]$; so the lemma follows.

We are now showing the second claim. By the algorithm, the fact that $s^*(t) = 0$ for some $t \in [r_j, d_j]$ means that job j will be processed at speed s^c in some interval $[a, b] \subset [r_j, d_j]$ (assuming that no new job is released after r_j). The marginal increase in the energy is $P(s^c)(b - a)$ while p_j could be expressed as $s^c(b - a)$. Therefore, $\lambda_j = P(s^c) / s^c$. \square

Theorem 3 *The algorithm has competitive ratio at most $\max\{4, \alpha^\alpha\}$.*

Proof Let E_1^* be the dynamic energy of the algorithm schedule, i.e., $E_1^* = \int_0^\infty [P(s^*(t)) - P(0)] dt \leq \sum_j \lambda_j p_j / \beta$ due to the definition of λ_j 's and $0 < \beta \leq 1$. Moreover, let E_2^* be the static energy plus the wake-up energy of the algorithm, i.e., $E_2^* = \int_0^\infty P(0) F^*(t) dt + \frac{A}{2} \int_0^\infty |(F^*)'(t)| dt$. We will bound the Lagrangian dual objective.

By Lemma 5 (second statement), for every job j such that $s^*(t) = 0$ for some $t \in [r_j, d_j]$, $\lambda_j = \frac{P(s^c)}{s^c}$. By the definition of the critical speed, $\lambda_j \leq \frac{P(z)}{z}$ for any $z > 0$. Therefore,

$$\sum_j \int_{r_j}^{d_j} s_j(t) F(t) \left(\lambda_j - \frac{P(s(t))}{s(t)} \right) dt \leq 0 \quad (6)$$

where in the sum is taken over jobs j such that $s^*(t) = 0$ for some $t \in [r_j, d_j]$. Therefore,

$$\begin{aligned}
L_1(s, \lambda) &:= \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{d_j} s_j(t) F(t) \left(\lambda_j - \frac{P(s(t))}{s(t)} \right) \mathbb{1}_{\{s(t)>0\}} \mathbb{1}_{\{F(t)=1\}} dt \\
&\geq \beta E_1^* - \max_{s, F} \sum_j \int_{r_j}^{d_j} s_j(t) F(t) \left[\beta P'(s^*(t)) - \frac{P(s(t))}{s(t)} \right] \mathbb{1}_{\{s(t)>0\}} \mathbb{1}_{\{F(t)=1\}} \mathbb{1}_{\{s^*(t)>0\}} dt \\
&\geq \beta E_1^* - \max_s \int_0^\infty s(t) \left[\beta P'(s^*(t)) - \frac{P(s(t))}{s(t)} \right] \mathbb{1}_{\{s(t)>0\}} \mathbb{1}_{\{F(t)=1\}} \mathbb{1}_{\{s^*(t)>0\}} dt \\
&\geq \beta E_1^* - \int_0^\infty \left[\beta P'(s^*(t)) \bar{s}(t) - P(\bar{s}(t)) \right] \mathbb{1}_{\{s(t)>0\}} \mathbb{1}_{\{F(t)=1\}} \mathbb{1}_{\{s^*(t)>0\}} dt \\
&\geq \beta E_1^* - \frac{1}{2} \int_0^\infty \left[\beta P'(s^*(t)) \bar{s}(t) - P(\bar{s}(t)) \right] \mathbb{1}_{\{s^*(t)>0\}} dt \\
&\quad - \frac{1}{2} \int_0^\infty \left[\beta P'(s^*(t)) \bar{s}(t) - P(\bar{s}(t)) \right] \mathbb{1}_{\{F(t)=1\}} dt
\end{aligned}$$

where in the second line, the sum is taken over jobs j such that $s^*(t) > 0$ for all $t \in [r_j, d_j]$. The first inequality follows (6) and Lemma 5 (first statement). The second inequality holds since $F(t) \leq 1$ and $s(t) = \sum_j s_j(t)$. The third inequality is due to the first order derivative and $\bar{s}(t)$ is the solution of equation $P'(z(t)) = \beta P'(s^*(t))$. In fact $\bar{s}(t)$ maximizes function $s(t) \beta P'(s^*(t)) - P(s(t))$.

As the energy power function $P(z) = z^\alpha + g$ where $\alpha \geq 1$ and $g \geq 0$, $\bar{s}(t)^{\alpha-1} = \beta(s^*(t))^{\alpha-1}$. Therefore,

$$\begin{aligned}
L_1(s, \lambda) &\geq \beta E_1^* - \frac{1}{2} \int_0^\infty \left(\beta \alpha (s^*(t))^{\alpha-1} \bar{s}(t) - (\bar{s}(t))^\alpha - g \right) \mathbb{1}_{\{s^*(t)>0\}} dt \\
&\quad - \frac{1}{2} \int_0^\infty \left(\beta \alpha (s^*(t))^{\alpha-1} \bar{s}(t) - (\bar{s}(t))^\alpha - g \right) \mathbb{1}_{\{F(t)=1\}} dt \\
&= \beta E_1^* - \int_0^\infty (\alpha - 1) \beta^{\alpha/(\alpha-1)} (s^*(t))^\alpha dt + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t)=1\}} dt \\
&= \left[\beta - (\alpha - 1) \beta^{\alpha/(\alpha-1)} \right] E_1^* + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t)=1\}} dt
\end{aligned}$$

Choose $\beta = 1/\alpha^{\alpha-1}$, we have that

$$L(s, F, \lambda) \geq \frac{1}{\alpha^\alpha} E_1^* + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t)=1\}} dt + \frac{A}{2} \int_0^\infty |F'(t)| dt$$

In the following, we claim that

$$L_2(F) := \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t)=1\}} dt + \frac{A}{2} \int_0^\infty |F'(t)| dt \geq E_2^*/4$$

for any feasible solution (s, F) of the relaxation.

Consider the algorithm schedule. An *end-time* u is a moment in the schedule such that the machine switches from the idle state to the sleep state. Conventionally, the first end-time in the schedule is 0. Partition the time line into phases. A *phase* $[u, v)$ is a time interval such that u, v

are consecutive end-times. Observe that in a phase, the schedule has transition cost A and there is always a new job released in a phase (otherwise the machines would not switch to non-sleep state). We will prove the claim on every phase. In the following, we are interested in phase $[u, v)$ and whenever we mention $L_2(F)$, it refers to $\frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{F(t)=1\}} dt + \frac{A}{2} \int_u^v |F'(t)| dt$.

By the algorithm, the static energy of the schedule during the idle time is A , i.e., $\int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt = A$. Let (s, F) be an arbitrary feasible of solution of the relaxation.

If during $[u, v)$, the machine following solution (s, F) makes a transition from non-sleep state to sleep state or inversely then $L_2(F) \geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{A}{2}$. Hence

$$L_2(F) \geq \frac{1}{4} \left(\int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + A \right) = \frac{1}{4} E_2^*|_{[u,v]}.$$

If during $[u, v)$, the machine following solution (s, F) makes no transition (from non-sleep static to sleep state or inversely) then $F(t) = 1$ during $[u, v)$ in order to process jobs released in the phase. Therefore,

$$\begin{aligned} L_2(F) &\geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{F(t)=1\}} dt = \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{2} \int_u^v g dt \\ &\geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{4} \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + \frac{A}{4} \\ &\geq \frac{1}{4} \left(\int_u^v g \mathbb{1}_{\{s^*(t) > 0\}} dt + \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + A \right) = \frac{1}{4} E_2^*|_{[u,v]} \end{aligned}$$

where the second inequality follows the algorithm: as the machine switches to sleep state at time v , it means that the total idle duration in $[u, v)$ incurs a cost A .

In conclusion, the dual $L(s, F, \lambda) \geq E_1^*/\alpha^\alpha + E_2^*/4$ whereas the primal is $E_1^* + E_2^*$. Thus, the competitive ratio is at most $\max\{4, \alpha^\alpha\}$. \square

6 Minimizing Energy plus Weighted Flow-Time in Speed Scaling with Power Down Model

The problem. We consider the problem of minimizing the total weighted flow-time plus energy on a single in the general energy model. Again, the machine has a transition cost A from sleep state to active state. The power energy consumption of the machine at time t in its active state is $P(s(t)) = s(t)^\alpha + g$ where $\alpha \geq 1$ and $g \geq 0$ and $s(t)$ is the machine speed at time t . Recall that the *dynamic energy* is $\int_0^\infty s^\alpha(t) dt$ and the *static energy* is $\int_0^\infty g dt$ (where the integrals are taken during active periods). Jobs arrive over time, a job j is released at time r_j , has weight w_j and requires p_j units of processing volume if it is processed on machine i . A job could be processed preemptively, i.e., a job could be interrupted and resumed later. The *flow-time* of a job j is $C_j - r_j$ where C_j is the completion time of the job. At any time, the scheduler has to determine the state and the speed of every machine (if it is active) and also a policy how to execute jobs. The objective is to minimize the total weighted flow-time of all jobs plus the total energy (including the wake-up cost).

Formulation. Similar as the previous section, we make use of the properties of Heaviside step function and Dirac delta function to encode the machine states and the transition cost. Let $F(t)$

be a function indicating whether the machine i is in active state at time t , i.e., $F(t) = 1$ if the machine is active at t and $F(t) = 0$ if it is in the sleep state. Assume that the machine initially is in the sleep state. Then $A \int_0^{+\infty} |F'(t)| dt$ equals twice the transition cost of the machine. Let $s_j(t)$ be the variable that represents the speed of job j at time t . Let C_j be a variable representing the completion time of j . The problem could be relaxed as the following (non-convex) program.

$$\begin{aligned}
& \text{minimize} \quad \int_0^\infty 2P\left(\sum_j s_j(t)\right) F(t) dt + 2 \sum_j \left(\int_{r_j}^{C_j} s_j(t) F(t) dt \right) \frac{w_j}{p_j} (C_j - r_j) \\
& \quad \quad \quad + A \int_0^\infty |F'(t)| dt \\
& \text{subject to} \quad \int_{r_j}^{C_j} s_j(t) F(t) dt = p_j \quad \forall j \\
& \quad \quad \quad s_j(t) \geq 0 \quad \forall j, t \geq r_j \\
& \quad \quad \quad F(t) \in \{0, 1\} \quad \forall t.
\end{aligned} \tag{7}$$

The first constraints ensure that every job j must be completed by time C_j . In the objective function, the first and second terms represent twice the consumed energy and the total weighted flow-time, respectively. Note that in the second term, $\int_{r_j}^{C_j} s_j(t) F(t) dt = p_j$ by the constraints. The last term stands for twice the transition cost.

Preliminaries. We say that a job j is *pending* at time t if it is not completed, i.e., $r_j \leq t < C_j$. At time t , denote $q_j(t)$ the *remaining* processing volume of job j . The *total weight* of pending jobs at time t is denoted as $W(t)$. The *density* of a job j is $\delta_j = w_j/p_j$. Define the *critical speed* s^c of the machine as $\arg \min_{z \geq 0} P(z)/z$. As $P(z) = z^\alpha + g$, by the first order condition, s^c satisfies $(\alpha - 1)(s^c)^\alpha = g$.

6.1 The Algorithm.

We first describe the algorithm informally. In the speed scaling model, all previous algorithms explicitly or implicitly balance the weighted flow-time of jobs and the consumed energy to process such jobs. That could be done by setting the machine speed at any time t proportional to some function of the total weight of pending jobs (precisely, proportional to $W(t)^{1/\alpha}$ where $W(t)$ is the total weight of pending jobs). Our algorithm follows the same idea of balancing. However, in the general energy model, the algorithm would not be competitive if the speed is always set proportionally to $W(t)^{1/\alpha}$ since the static energy might be large due to the long active periods of the machine. Hence, even if the total weight of pending jobs on the machine is small, in some situation the speed is maintained larger than $W(t)^{1/\alpha}$. In fact, it will be set to be the critical speed s^c , defined as $\arg \min P(z)/z$.

An issue while dealing with the general model is to determine when a machine is waken up. Again, if the total weight of pending jobs is small and the machine is active, then the static energy is large. Otherwise if pending jobs remain for long time then the weight flow-time is large. The algorithm also balances the costs by making a plan and switching the machine into active state at appropriate moments. If new job is released then the plan, together with its starting time, will be changed.

Description of algorithm. At any time t , the machine maintains the following policy in different states: the *working state* (the machine is active and currently processes some job), the *idle state* (the machine is active but currently processes no job) and the *sleep state*.

In working state. If $\frac{\alpha}{\alpha-1}W(t)^{(\alpha-1)/\alpha} > P(s^c)/s^c$ then the machine speed is set as $W(t)^{1/\alpha}$. Otherwise, the speed is set as s^c . At any time, the machine processes the highest density job among the pending ones.

In idle state. 1. If $\frac{\alpha}{\alpha-1}W(t)^{(\alpha-1)/\alpha} > P(s^c)/s^c$ then switch to the working state.

2. If $0 < \frac{\alpha}{\alpha-1}W(t)^{\frac{\alpha-1}{\alpha}} \leq P(s^c)/s^c$ then make a plan to process the pending jobs with speed (exactly) s^c in non-increasing order of their density. So the plan consists of a single block (with no idle time) and the block length could be explicitly computed (given the processing volumes of all jobs and speed s^c). Hence, the total energy consumed in the plan could also be computed and it is independent of the starting time of the plan.

Choose the starting time of the plan in such a way that the total energy consumed in the plan equals the total weighted flow-time of all jobs in the plan. There always exists such starting time since if the plan begins immediately at t , the energy is larger than the weighted flow-time; and inversely if the starting time is large enough, the latter dominates the former.

At the starting time of a plan, switch to the working state. (Note that the plan together with its starting time could be changed due to the arrival of new jobs.)

3. Otherwise, if the total duration of idle state from the last wake-up equals A/g then switch to sleep state.

In sleep state. Use the same policy as the first two steps of the idle state.

6.2 Analysis

The Lagrangian dual of program (7) is $\max \min_{x,s,C,F} L$ where L is the corresponding Lagrangian function where the maximum is taken over dual variables. The purpose of the section is to choose appropriate dual variables and prove that for any feasible solution (x, s, C, F) of the primal, the Lagrangian dual is bounded by a desired factor from the primal.

Dual variables. Denote the dual variables corresponding to the first constraints of (7) as λ_j 's. Set all dual variables (corresponding to the primal (7)) except λ_j 's equal to 0. The values of dual variables λ_j 's is defined as the follows.

Fix a job j . At the arrival of a job j , rename pending jobs as $\{1, \dots, k\}$ in non-increasing order of their densities, i.e., $p_1/w_1 \leq \dots \leq p_k/w_k$ (note that p_a/w_a is the inverse of job a 's density). Denote $W_a = w_a + \dots + w_k$ for $1 \leq a \leq k$.

Define λ_j such that

$$\lambda_j p_j = w_j \sum_{a=1}^j \frac{q_a(r_j)}{W_a^{1/\alpha}} + W_{j+1} \frac{q_j(r_j)}{W_j^{1/\alpha}} + P(s^c) \frac{q_j(r_j)}{s^c} \quad (8)$$

Note that $q_j(r_j) = p_j$. If job j is processed with speed larger than s^c then the first term stands for the weighted flow-time of j and the second term represents an upper bound of the increase in the

weighted flow-time of jobs with density smaller than δ_j . Observe that due to arrival of j , the jobs with higher density than δ_j are completed earlier and the ones with smaller density than δ_j may have higher flow-time. Informally, the second sum in (8) captures the marginal change in the total weighted flow-time. The third term in (8) is introduced in order to cover energy consumed during the execution periods of job j if it is processed by speed s^c . That term is necessary since during such periods the energy consumption and the weighted flow-time is not balanced.

The Lagrangian function $L(x, s, C, F, \lambda)$ with the chosen dual variables becomes

$$\begin{aligned}
& A \int_0^\infty |F'(t)| dt + 2 \int_0^\infty P\left(\sum_j s_j(t)\right) F(t) dt + 2 \sum_j \delta_j (C_j - r_j) \int_{r_j}^{C_j} s_j(t) F(t) dt \\
& \quad + \sum_j \lambda_j \left(p_j - \int_{r_j}^{C_j} s_j(t) F(t) dt \right) \\
& = \sum_j \lambda_j p_j + A \int_0^\infty |F'(t)| dt + \sum_j \int_{r_j}^{C_j} \delta_j (C_j - r_j) s_j(t) F(t) dt \\
& \quad - \sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j (C_j - r_j) \right) dt
\end{aligned}$$

Notations. We denote $s^*(t)$ the machine speed at time t by the algorithm. So by the algorithm, if $s^*(t) > 0$ then $s^*(t) \geq s^c$. Let \mathcal{E}_1^* and \mathcal{E}_2^* be the total dynamic and static energy consumed by the algorithm schedule, respectively. In other words, $\mathcal{E}_1^* = \int_0^\infty (s^*(t))^\alpha dt$ and $\mathcal{E}_2^* = \int_0^\infty g$ where the integral is taken over all moments t where the machine is active (either in working or in idle states). Additionally, let \mathcal{E}_3^* be the total transition cost of the machine. Moreover, let \mathcal{F}^* be the total weighted flow-time of all jobs in the schedule.

We relate the cost of the schedule (due to the algorithm) and the chosen values of dual variables by the following lemma. Note that by definition of λ_j 's, we have that $\sum_j \lambda_j p_j \geq \mathcal{F}^*$.

Lemma 6 *It holds that $2\mathcal{E}_1^* + 3\mathcal{E}_2^* \geq \mathcal{F}^*$ and $\sum_j \lambda_j p_j \geq \mathcal{E}_1^*$.*

Proof We prove the first inequality. Consider times t where the machine speed is s^c . By the algorithm $P(s^c)/s^c \geq \frac{\alpha}{\alpha-1} W(t)^{(\alpha-1)/\alpha}$. Recall that by the definition of critical speed $g = (\alpha - 1)(s^c)^\alpha$, so $\alpha(s^c)^\alpha = P(s^c)$. Therefore, $s^c \geq (\alpha - 1)^{-1/(\alpha-1)} W(t)^{1/\alpha}$. Hence,

$$\begin{aligned}
2\mathcal{E}_2^* & \geq (\alpha - 1)^{\frac{1}{\alpha-1}} \int_0^\infty g \mathbb{1}_{\{s^*(t)=s^c\}} dt = (\alpha - 1)^{\frac{1}{\alpha-1}} \int_0^\infty (\alpha - 1)(s^c)^\alpha \mathbb{1}_{\{s^*(t)=s^c\}} dt \\
& \geq \int_0^\infty W(t) \mathbb{1}_{\{s^*(t)=s^c\}} dt
\end{aligned}$$

Now consider times t where the machine speed is $W(t)^{1/\alpha}$ strictly larger than s^c . Thus the dynamic energy consumed on such periods is

$$\mathcal{E}_1^* \geq \int_0^\infty (s^*(t))^\alpha \mathbb{1}_{\{s^*(t) > s^c\}} dt \geq \int_0^\infty W(t) \mathbb{1}_{\{s^*(t) > s^c\}} dt$$

For periods where $s^*(t) = 0$ while some jobs are still pending on the machine, by the algorithm plan, the total weighted flow time of jobs in such periods is bounded by $(\mathcal{E}_1^* + \mathcal{E}_2^*)$. Therefore,

$$2\mathcal{E}_1^* + 3\mathcal{E}_2^* \geq \int_0^\infty W(t)dt = \mathcal{F}^*.$$

In the rest, we prove the second inequality $\sum_j \lambda_j p_j \geq \mathcal{E}_1^*$. By the definition of λ_j 's (particularly the third term in (8)), $\sum_j \lambda_j p_j$ covers the total energy of machine during all intervals where the machine processes jobs by speed s^c . Denote Γ as $\sum_j \lambda_j p_j$ subtracting the energy incurred during periods where the machine speed is s^c . We need to prove that Γ is enough to cover the total energy incurred over all moments where the machine speed is strictly larger than s^c . In the following, we are interested only in such moments.

Consider a job k processed at time t with speed larger than s^c . By the definition of λ_j 's, Γ contributes to time t an amount at least $\sum_j w_j / W(t)^{1/\alpha}$ where the sum is taken over pending jobs j with smaller density than that of k . The latter is exactly $W(t)$. Thus, Γ contributes to time t an amount $W(t)^{(\alpha-1)/\alpha}$.

Now consider an arbitrarily small interval $[a, b]$ where the machine processes only job k and the speed is strictly larger than s^c . Let W be the total weight of pending jobs over $[a, b]$. The processing amount of k done over $[a, b]$ is $W^{1/\alpha}(b-a)$ while the energy amount consumed in that interval is $W(b-a)$. Hence, in average the machine spends $W^{(\alpha-1)/\alpha}$ (dynamic) energy unit at time t .

Therefore, during periods where the machine speed is larger than s^c , Γ is increase at rate proportionally to the one of the dynamic energy. The second inequality of the lemma follows. \square

Corollary 1 *It holds that $\sum_j \lambda_j p_j \geq \frac{7}{8}\mathcal{E}_1^* + \frac{1}{16}\mathcal{F}^* - \frac{3}{16}\mathcal{E}_2^*$.*

Proof By the previous lemma, we deduce that

$$\sum_j \lambda_j p_j \geq \mathcal{E}_1^* \geq \frac{7}{8}\mathcal{E}_1^* + \frac{1}{8}\left(\frac{1}{2}\mathcal{F}^* - \frac{3}{2}\mathcal{E}_2^*\right) = \frac{7}{8}\mathcal{E}_1^* + \frac{1}{16}\mathcal{F}^* - \frac{3}{16}\mathcal{E}_2^*.$$

\square

In the following, we show the main technical lemma.

Lemma 7 *Let j be an arbitrary job. Then, for every $t \geq r_j$*

$$\lambda_j - \delta_j(t - r_j) \leq \max\left\{\frac{\alpha}{\alpha-1}W(t)^{\frac{\alpha-1}{\alpha}} + \frac{P(s^c)}{s^c}, 2\frac{P(s^c)}{s^c}\right\} \quad (9)$$

Proof Fix a job j . We prove by induction on the number of released jobs after r_j . The base case follows Lemma 8 and the induction step is done by Lemma 9. \square

Lemma 8 *If no new job is released after r_j then inequality (9) holds.*

Proof Denote the instance as \mathcal{I}_0 . At r_j , rename jobs in non-increasing order of their densities, i.e., $p_1/w_1 \leq \dots \leq p_n/w_n$ (note that p_a/w_a is the inverse of job a 's density). Denote $W_a = w_a + \dots + w_n$ for $1 \leq a \leq n$.

By definition of λ_j ,

$$\lambda_j - \frac{P(s^c)}{s^c} = \delta_j \left[\frac{q_1(r_j)}{W_1^{1/\alpha}} + \dots + \frac{q_j(r_j)}{W_j^{1/\alpha}} \right] + \frac{W_{j+1}}{W_j^{1/\alpha}}$$

Let $C_a(\mathcal{I}_0)$ be the completion time of job a for every a . Moreover, let ℓ be the largest job index such that $\frac{\alpha}{\alpha-1} W_\ell^{(\alpha-1)/\alpha} \geq P(s^c)/s^c$. In other words, job ℓ is processed by speed strictly larger than s^c and the other jobs with larger index (if exist) will be processed by speed s^c . Fix a time t , let k be the pending job at t with the smallest index. We prove first the following claim.

Claim 1 *It holds that*

$$\lambda_j - \delta_j(t - r_j) - \frac{P(s^c)}{s^c} \leq \max \left\{ \frac{w_k}{W_k^{1/\alpha}} + \frac{w_{k+1}}{W_{k+1}^{1/\alpha}} + \dots + \frac{w_n}{W_n^{1/\alpha}}, \frac{P(s^c)}{s^c} \right\}$$

Proof of claim We consider different cases.

Case 1: $\ell \leq j$. In this case, job j will be processed by speed s^c .

Subcase 1.1: $t \leq C_\ell(\mathcal{I}_0)$. During interval $[r_j, t]$, the machine has completed jobs $1, \dots, k-1$ and has processed a part of job k . Precisely, during $[r_j, t]$ the machine has processed $q_a(r_j)$ units of job a for every job $1 \leq a < k$ and has executed $(q_k(r_j) - q_k(t))$ units of job k . Moreover, every job $1 \leq a \leq k$ is processed with speed $W_a^{1/\alpha}$. Therefore,

$$\begin{aligned} \lambda_j - \delta_j(t - r_j) - \frac{P(s^c)}{s^c} &= \delta_j \left[\frac{q_k(t)}{W_k^{1/\alpha}} + \frac{q_{k+1}(r_j)}{W_{k+1}^{1/\alpha}} + \dots + \frac{q_j(r_j)}{W_j^{1/\alpha}} \right] + \frac{W_{j+1}}{W_j^{1/\alpha}} \\ &\leq \delta_j \left[\frac{p_k}{W_k^{1/\alpha}} + \frac{p_{k+1}}{W_{k+1}^{1/\alpha}} + \dots + \frac{p_j}{W_j^{1/\alpha}} \right] + \frac{W_{j+1}}{W_j^{1/\alpha}} \\ &= \delta_j \left(\frac{w_k}{\delta_k W_k^{1/\alpha}} + \frac{w_{k+1}}{\delta_{k+1} W_{k+1}^{1/\alpha}} + \dots + \frac{w_j}{\delta_j W_j^{1/\alpha}} \right) + \frac{W_{j+1}}{W_j^{1/\alpha}} \\ &\leq \frac{w_k}{W_k^{1/\alpha}} + \frac{w_{k+1}}{W_{k+1}^{1/\alpha}} + \dots + \frac{w_j}{W_j^{1/\alpha}} + \frac{w_{j+1}}{W_{j+1}^{1/\alpha}} + \dots + \frac{w_n}{W_n^{1/\alpha}} \\ &\leq \int_0^{W(t)} \frac{dz}{z^{1/\alpha}} = \frac{\alpha}{\alpha-1} W(t)^{\frac{\alpha-1}{\alpha}} \end{aligned}$$

The first inequality is because $q_a(r_j) \leq p_a$ for every job a . The first equality is due to the definition of the density. The second inequality follows since $\delta_j \leq \delta_a$ for every job $a \leq j$ and $W_{j+1} \geq \dots \geq W_n$. The third inequality holds since function $z^{-1/\alpha}$ is decreasing.

Subcase 1.2: $t > C_\ell(\mathcal{I}_0)$. In this case $k > \ell$, i.e., during $[r_j, t]$ the machine i has completed jobs $1, \dots, \ell$. Similarly as the previous subcase, we have

$$\begin{aligned} \lambda_j - \delta_j(t - r_j) - \frac{P(s^c)}{s^c} &\leq \sum_{a=\ell+1}^n \frac{w_a}{W_a^{1/\alpha}} \\ &\leq \int_0^{W_{\ell+1}} \frac{dz}{z^{1/\alpha}} = \frac{\alpha}{\alpha-1} W_{\ell+1}^{\frac{\alpha-1}{\alpha}} \leq \frac{P(s^c)}{s^c} \end{aligned}$$

where the last inequality follows the definition of ℓ .

Case 2: $\ell > j$. In this case, job j will be processed with speed strictly larger than s^c .

Subcase 2.1: $t \leq C_j(\mathcal{I}_0)$. The proof is done in the same manner as in Subcase 1.1.

Subcase 2.2: $t > C_j(\mathcal{I}_0)$. For simplicity, denote $C_a = C_j(\mathcal{I}_0)$. Partition time after $C_j(\mathcal{I}_0)$ as $\cup_{a=j}^n [C_a, C_{a+1})$. During an interval $[C_a, C_{a+1})$, the weight W_a is unchanged so to show inequality (9), it is sufficient to prove at $t = C_j, C_{j+1}, \dots, C_{n-1}$.

We prove again by induction. For the base case $t = C_j$, the claim inequality holds by the previous case. Assume that the inequality holds at $t = C_a$, we will prove that it holds at $t = C_{a+1}$ for $a \geq j$. We are interested only in $\tau \in [C_a, C_{a+1})$. Let $V(\tau) = w_a q_a(\tau)/p_a + w_{a+1} + \dots + w_n$. Informally, $V(\tau)$ is the fractional weight of pending jobs at time τ .

During period $[\tau, \tau + d\tau]$ assume that the total fractional weight of pending jobs varies by $dV(\tau)$. During the same period $[\tau, \tau + d\tau]$, the total processing volume done by algorithm is at least $W_a^{1/\alpha} d\tau$ since the speed is either $W(\tau)^{1/\alpha} (= W_a^{1/\alpha})$ or s^c but in the latter, by the algorithm, $s^c \geq W(\tau)^{1/\alpha}$. Moreover, jobs a processed during $[\tau, \tau + d\tau]$ have density at most δ_j . Therefore, $dV(\tau) \leq \delta_j W_a^{1/\alpha} d\tau$. In other words, $V'(\tau) d\tau \leq \delta_j W_a^{1/\alpha} d\tau$. Taking integral, we get

$$w_a = W_a - W_{a+1} = \int_{C_a}^{C_{a+1}} V'(\tau) d\tau \leq \int_{C_a}^{C_{a+1}} \delta_j W_a^{1/\alpha} d\tau = \delta_j W_a^{1/\alpha} (C_{a+1} - C_a) \quad (10)$$

Therefore,

$$\begin{aligned} \lambda_j - \delta_j(C_{a+1} - r_j) - \frac{P(s^c)}{s^c} &= \lambda_j - \delta_j(C_a - r_j) - \frac{P(s^c)}{s^c} - \delta_j(C_{a+1} - C_a) \\ &\leq \max \left\{ \frac{w_a}{W_a^{1/\alpha}} + \dots + \frac{w_n}{W_n^{1/\alpha}}, \frac{P(s^c)}{s^c} \right\} - \frac{w_a}{W_a^{1/\alpha}} \\ &\leq \max \left\{ \frac{w_{a+1}}{W_{a+1}^{1/\alpha}} + \dots + \frac{w_n}{W_n^{1/\alpha}}, \frac{P(s^c)}{s^c} \right\} \end{aligned}$$

where the first inequality is due to the induction hypothesis and inequality (10).

Combining all the cases, the claim holds. \square

Using the claim, the lemma follows immediately as shown below.

$$\begin{aligned} \lambda_j - \delta_j(t - r_j) - \frac{P(s^c)}{s^c} &\leq \max \left\{ \frac{w_k}{W_k^{1/\alpha}} + \dots + \frac{w_n}{W_n^{1/\alpha}}, \frac{P(s^c)}{s^c} \right\} \\ &\leq \max \left\{ \int_0^{W(t)} \frac{dz}{z^{1/\alpha}}, \frac{P(s^c)}{s^c} \right\} = \max \left\{ \frac{\alpha}{\alpha - 1} W(t)^{\frac{\alpha-1}{\alpha}}, \frac{P(s^c)}{s^c} \right\} \end{aligned}$$

where the inequality holds since function $z^{-1/\alpha}$ is decreasing. (Recall that k is the pending job at time t with the smallest index.) \square

Lemma 9 Assume that inequality (9) holds if there are $(n - 1)$ jobs released after r_j . Then the inequality also holds if n jobs are released after r_j .

Proof Denote the instance as \mathcal{I}_n . Among such jobs, let n be the last released one (at time r_n). By induction hypothesis, it remains to prove the lemma inequality for $t \geq r_n$.

We first show the claim that inequality (9) holds for any time $t \geq C_j(\mathcal{I}_n)$ by a similar argument as in Subcase 2.2 of the previous claim. Indeed, we prove the claim by fixing the processing volume of job n and varying its weight w_n . Note that $C_j(\mathcal{I}_n)$ depends on w_n and when w_n is varied, $C_j(\mathcal{I}_n)$ is also varied. However, with a fixed value of w_n , $C_j(\mathcal{I}_n)$ is fixed and we are interested only in $t \geq C_j(\mathcal{I}_n)$. If $w_n = 0$ then the claim follows the induction hypothesis (the instance becomes the one with $(n-1)$ jobs). Assume that the claim holds for some value w_n . Now increase an arbitrarily small amount of w_n and consider a time $t \geq C_j(\mathcal{I}_n)$ (corresponding to the current value of w_n). Due to that increase, during period $[t, t+dt]$ the total fractional weight of pending jobs varies by $dV(t)$. During the same period $[t, t+dt]$, the total processing volume done by algorithm is at least $V(t)^{1/\alpha}dt$ since the machine speed is at least $W(t)^{1/\alpha}$. Moreover, jobs processed during $[t, t+dt]$ have density at most δ_j . Therefore, $dV(t) \leq \delta_j V(t)^{1/\alpha}dt$. So

$$\frac{\alpha}{\alpha-1}dW(t)^{(\alpha-1)/\alpha} = \frac{dW(t)}{W(t)^{1/\alpha}} \leq \delta_j dt$$

This inequality means that in the lemma inequality (9), the decrease in the left-hand side is larger than that in the right-hand side while varying the weight of job n . Hence, the inequality holds for $t \geq C_j(\mathcal{I}_n)$.

Now we consider instance \mathcal{I}_n with fixed parameters for job n . We will prove the lemma for $t < C_j(\mathcal{I}_n)$. Denote $t_0 = C_j(\mathcal{I}_n)$. Again, rename jobs in non-increasing order of their densities at time r_n . (After r_n , no new job is released and the relative order of jobs is unchanged.) Let W_a be the total weight of pending jobs at r_n which have density smaller than δ_a . Recall that the total weight of pending jobs at time t is $W(t)$.

Let k be the pending job with the smallest index at time t in the instance \mathcal{I}_n . During $[t, t_0]$, the machine processes (a part) of job k , jobs $k+1, \dots, j$. The jobs have density at least δ_j . We deduce

$$\begin{aligned} \lambda_j - \delta_j(t - r_j) - \frac{P(s^c)}{s^c} &= \lambda_j - \delta_j(t_0 - r_j) - \frac{P(s^c)}{s^c} + \delta_j(t_0 - t) \\ &\leq \frac{\alpha}{\alpha-1}W(t_0)^{\frac{\alpha-1}{\alpha}} + \delta_j(t_0 - t) \\ &\leq \frac{\alpha}{\alpha-1}W(t_0)^{\frac{\alpha-1}{\alpha}} + \delta_j \left(\frac{q_k(t)}{W_k^{1/\alpha}} + \frac{q_{k+1}(r_n)}{W_{k+1}^{1/\alpha}} + \dots + \frac{q_j(r_n)}{W_j^{1/\alpha}} \right) \\ &\leq \frac{\alpha}{\alpha-1}W(t_0)^{\frac{\alpha-1}{\alpha}} + \int_{W_{j+1}}^{W_k} \frac{dz}{z^{1/\alpha}} \\ &\leq \frac{\alpha}{\alpha-1}W_k^{\frac{\alpha-1}{\alpha}} = \frac{\alpha}{\alpha-1}W(t)^{\frac{\alpha-1}{\alpha}} \end{aligned}$$

The first inequality follows the previous claim, stating that inequality (9) holds for $t \geq t_0$. The second inequality follows the fact that at any time the speed of the machine is at least $W(t)^{1/\alpha}$. The third inequality holds since $\delta_k \leq \delta_{k+1} \leq \dots \leq \delta_j$ and function $z^{-1/\alpha}$ is decreasing. The last inequality holds since $W(t_0) = W_{j+1}$ and $W_k = W(t)$. \square

Theorem 4 *The algorithm has competitive ratio at most $\max\{64, 32\alpha/\ln \alpha\}$.*

Proof Recall that the dual has value at least $\min L(x, s, C, F, \lambda)$ where the minimum is taken over (x, s, C, F) feasible solution of the primal. The goal is to bound the Lagrangian function.

$$\begin{aligned} L(x, s, C, F, \lambda) &= \sum_j \lambda_j p_j + A \int_0^\infty |F'(t)| dt + \sum_j \int_{r_j}^{C_j} \delta_j (C_j - r_j) s_j(t) F(t) dt \\ &\quad - \sum_{i,j} \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j (C_j - r_j) \right) \mathbf{1}_{\{s(t) > 0\}} dt \end{aligned} \quad (11)$$

Define $L_1(x, s, C, F, \lambda)$ as

$$\sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j (C_j - r_j) \right) \mathbf{1}_{\{s(t) > 0\}} dt$$

Claim 2 *Let (x, s, C, F) be an arbitrary feasible solution of the primal. Then,*

$$L_1(x, s, C, F, \lambda) \leq \frac{\alpha - 1}{(\alpha - 1)^{\frac{\alpha}{\alpha-1}}} \mathcal{F}^* - \frac{1}{2} \int_0^\infty g \mathbf{1}_{\{F(t) > 0\}} dt - \frac{1}{2} \int_0^\infty g \mathbf{1}_{\{s^*(t) > 0\}} dt$$

Claim 3 *Let (x, s, C, F) be an arbitrary feasible solution of the primal. Define*

$$\begin{aligned} L_2(F) &:= \sum_j \int_{r_j}^{C_j} \delta_j (C_j - r_j) s_j(t) F(t) dt + A \int_0^\infty |F'(t)| dt \\ &\quad + \frac{1}{2} \int_0^\infty g \mathbf{1}_{\{F(t) > 0\}} dt + \frac{1}{2} \int_0^\infty g \mathbf{1}_{\{s^*(t) > 0\}} dt \end{aligned}$$

Then, $L_2(F) \geq \mathcal{E}_2^/4$.*

We first show how to prove the theorem assuming the claims. By (11), we have

$$\begin{aligned} L(x, s, C, F, \lambda) &\geq \sum_j \lambda_j p_j + A \int_0^\infty |F'(t)| dt - \sum \frac{\alpha - 1}{(\alpha - 1)^{\frac{\alpha}{\alpha-1}}} \mathcal{F}^* \\ &\quad + \frac{1}{2} \int_0^\infty g \mathbf{1}_{\{F(t) > 0\}} dt + \frac{1}{2} \int_0^\infty g \mathbf{1}_{\{s^*(t) > 0\}} dt \\ &\geq \sum_j \lambda_j p_j - \frac{\alpha - 1}{(\alpha - 1)^{\frac{\alpha}{\alpha-1}}} \mathcal{F}^* + \frac{1}{4} \mathcal{E}_2^* + \frac{1}{4} \mathcal{E}_3^* \\ &\geq \left(1 - \frac{1}{(\alpha - 1)^{1/(\alpha-1)}} \right) \left(\frac{7}{8} \mathcal{E}_1^* + \frac{1}{16} \mathcal{F}^* - \frac{3}{16} \mathcal{E}_2^* \right) + \frac{1}{4} \mathcal{E}_2^* + \frac{1}{4} \mathcal{E}_3^* \\ &= \left(1 - \frac{1}{(\alpha - 1)^{1/(\alpha-1)}} \right) \left(\frac{7}{8} \mathcal{E}_1^* + \frac{1}{16} \mathcal{F}^* \right) + \left(\frac{1}{4} - \frac{3}{16} \right) \mathcal{E}_2^* + \frac{1}{4} \mathcal{E}_3^* \\ &\geq \frac{\ln(\alpha - 1)}{(\alpha - 1)^{\alpha/(\alpha-1)}} \left(\frac{3}{4} \mathcal{E}_1^* + \frac{1}{8} \mathcal{F}^* \right) + \frac{1}{16} \mathcal{E}_2^* + \frac{1}{4} \mathcal{E}_3^* \end{aligned}$$

where the first and second inequalities are due to Claim 2 and Claim 3, respectively. The third inequality follows Corollary 1 and $\sum_j \lambda_j p_j \geq \mathcal{F}^*$. The last inequality is due to the fact that $(\alpha - 1)^{\frac{1}{\alpha-1}} \geq 1 + \frac{\ln(\alpha-1)}{\alpha-1}$ for every $\alpha > 1$.

Besides, the primal objective is at most $2(\mathcal{F}^* + \mathcal{E}_1^* + \mathcal{E}_2^* + \mathcal{E}_3^*)$. Hence, the competitive ratio is at most $\max\{32\alpha/\ln \alpha, 64\}$.

In the rest, we prove the claims.

Claim 2 *Let (x, s, C, F) be an arbitrary feasible solution of the primal. Then,*

$$L_1(x, s, C, F, \lambda) \leq \frac{\alpha - 1}{(\alpha - 1)^{\frac{\alpha}{\alpha-1}}} \mathcal{F}^* - \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t) > 0\}} dt - \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t) > 0\}} dt$$

Proof of claim We have

$$\begin{aligned} L_1(x, s, C, F, \lambda) &:= \sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j(C_j - r_j) \right) \mathbb{1}_{\{s(t) > 0\}} dt \\ &\leq \sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j(t - r_j) \right) \mathbb{1}_{\{s(t) > 0\}} dt \end{aligned}$$

where the inequality holds because the integral for each job j is taken over $r_j \leq t \leq C_j$.

Let T be the set of time t such that $\frac{\alpha}{(\alpha-1)} W(t)^{\frac{\alpha-1}{\alpha}} \leq \frac{P(s^c)}{s^c}$. Then by Lemma 7, for any $t \in T$

$$\lambda_j - \delta_j(t - r_j) \leq 2 \frac{P(s^c)}{s^c}$$

Therefore,

$$\sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j(C_j - r_j) \right) \mathbb{1}_{\{t \in T\}} dt \leq 0 \quad (12)$$

since $s^c = \arg \min_{z \geq 0} P(z)/z$. Hence,

$$\begin{aligned}
& L_1(x, s, C, F, \lambda) \\
& \leq \sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2 \frac{P(s(t))}{s(t)} - \delta_j(t - r_j) \right) \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) > 0\}} \mathbb{1}_{\{t \notin T\}} dt \\
& \leq \sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\frac{\alpha}{(\alpha - 1)} W(t)^{\frac{\alpha-1}{\alpha}} - \frac{P(s(t))}{s(t)} \right) \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) > 0\}} \mathbb{1}_{\{t \notin T\}} dt \\
& = \int_0^\infty s(t) \left(\frac{\alpha}{(\alpha - 1)} W(t)^{\frac{\alpha-1}{\alpha}} - \frac{P(s(t))}{s(t)} \right) \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) > 0\}} \mathbb{1}_{\{t \notin T\}} dt \\
& \leq \int_0^\infty \left(\frac{\alpha}{(\alpha - 1)} W(t)^{\frac{\alpha-1}{\alpha}} \bar{s}(t) - P(\bar{s}(t)) \right) \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) > 0\}} \mathbb{1}_{\{t \notin T\}} dt \\
& \leq \int_0^\infty \left(\frac{\alpha}{(\alpha - 1)} W(t)^{\frac{\alpha-1}{\alpha}} \bar{s}(t) - P(\bar{s}(t)) \right) \mathbb{1}_{\{F(t) > 0\}} \mathbb{1}_{\{s^*(t) > 0\}} dt \\
& \leq \frac{1}{2} \int_0^\infty \left(\frac{\alpha}{(\alpha - 1)} W(t)^{\frac{\alpha-1}{\alpha}} \bar{s}(t) - P(\bar{s}(t)) \right) \mathbb{1}_{\{F(t) > 0\}} dt \\
& \quad + \frac{1}{2} \int_0^\infty \left(\frac{\alpha}{(\alpha - 1)} W(t)^{\frac{\alpha-1}{\alpha}} \bar{s}(t) - P(\bar{s}(t)) \right) \mathbb{1}_{\{s^*(t) > 0\}} dt
\end{aligned}$$

The first inequality is due to (12) and note that if $F(t) = 0$ then the contribution of the term inside the integral is 0. The second inequality follows Lemma 7 and recall that $s^c = \arg \min_{z \geq 0} P(z)/z$. The equality is because $\sum_j s_j(t) F(t) = \sum_j s_j(t) = s(t)$ for t such that $F(t) > 0$ (meaning $F(t) = 1$). The third inequality is due to the first order derivative and $\bar{s}(t)$ is the solution of $P'(z) = \frac{\alpha}{(\alpha-1)} W(t)^{\frac{\alpha-1}{\alpha}}$. The fourth inequality holds since the term inside the integral is non-negative and $\{t : t \notin T\} \subset \{t : s^*(t) > s^c\} \subset \{t : s^*(t) > 0\}$.

Replacing $\bar{s}(t) = (\alpha - 1)^{-1/(\alpha-1)} W(t)^{1/\alpha}$ (solution of $P'(z) = \frac{\alpha}{(\alpha-1)} W(t)^{\frac{\alpha-1}{\alpha}}$), we get:

$$\begin{aligned}
L_1(x, s, C, F, \lambda) & \leq \frac{\alpha - 1}{(\alpha - 1)^{\frac{\alpha}{\alpha-1}}} \int_0^\infty W(t) - \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t) > 0\}} dt - \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t) > 0\}} dt \\
& = \frac{\alpha - 1}{(\alpha - 1)^{\frac{\alpha}{\alpha-1}}} \mathcal{F}^* - \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t) > 0\}} dt - \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t) > 0\}} dt
\end{aligned}$$

□

Claim 3 Let (x, s, C, F) be an arbitrary feasible solution of the primal. Define

$$\begin{aligned}
L_2(F) & := \sum_j \int_{r_j}^{C_j} \delta_j(C_j - r_j) s_j(t) F(t) dt + A \int_0^\infty |F'(t)| dt \\
& \quad + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{F(t) > 0\}} dt + \frac{1}{2} \int_0^\infty g \mathbb{1}_{\{s^*(t) > 0\}} dt
\end{aligned}$$

Then, $L_2(F) \geq (\mathcal{E}_2^* + \mathcal{E}_3^*)/4$.

Proof of claim Consider the algorithm schedule. An *end-time* u is a moment in the algorithm schedule such that the machine switches from the idle state to the sleep state. Conventionally,

the first end-time in the schedule is 0. Partition the time line into phases. A *phase* $[u, v)$ is a time interval such that u, v are consecutive end-times. Observe that in a phase, the schedule has transition cost A and some new job is released in a phase (otherwise the machine would not switch to non-sleep state). We will prove the claim on every phase. In the following, we are only interested in phase $[u, v)$ and define

$$\begin{aligned} L_2(F)|_{[u,v)} &:= \sum_{j:u \leq r_j < v} \int_{r_j}^{C_j} \delta_j(C_j - r_j) s_j(t) F(t) dt + A \int_0^\infty |F'(t)| dt \\ &\quad + \frac{1}{2} \int_u^v g \mathbb{1}_{\{F(t)>0\}} dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt \end{aligned}$$

By the algorithm, the static energy on machine i during its idle time is A , i.e., $\int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt = A$. If during $[u, v)$, the schedule induced by solution (x, s, C, F) makes a transition from non-sleep state to sleep state or inversely then $L_2(F)|_{[u,v)} \geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + A$. Hence

$$L_2(F)|_{[u,v)} \geq \frac{1}{2} \left(\int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + A \right) = \frac{1}{2} \mathcal{E}_2^*|_{[u,v)} + \frac{1}{2} \mathcal{E}_3^*|_{[u,v)}.$$

If during $[u, v)$, the schedule induced by solution (x, s, C, F) makes no transition (from non-sleep static to sleep state or inversely) then either $F(t) = 1$ or $F(t) = 0$ for every $t \in [u, v]$. Note that by definition of phases, some job is released during $[u, v)$.

Case 1: $F(t) = 1 \ \forall u \leq t \leq v$. Hence,

$$\begin{aligned} L_2(F)|_{[u,v)} &\geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{F(t)=1\}} dt = \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{2} \int_u^v g dt \\ &\geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{1}{4} \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + \frac{A}{4} \\ &\geq \frac{1}{4} \left(\int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + A \right) = \frac{1}{4} \mathcal{E}_2^*|_{[u,v)} + \frac{1}{4} \mathcal{E}_3^*|_{[u,v)} \end{aligned}$$

where the second inequality follows since the total idle duration in $[u, v)$ incurs a cost A (so the machine switches to sleep state at time v).

Case 2: $F(t) = 0 \ \forall u \leq t \leq v$. As the machine is in the sleep state during $[u, v)$ in solution (x, s, C, F) , all jobs released in $[u, v)$ are completed later than v . By the algorithm, the total weighted flow-time of such jobs is at least the static energy of the algorithm during $[u, v)$. In other words,

$$\begin{aligned} L_2(F)|_{[u,v)} &\geq \sum_{j:u \leq r_j < v} \int_{r_j}^{C_j} \delta_j(C_j - r_j) s_j(t) F(t) dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt \\ &\geq \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt \\ &\geq \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)=0\}} dt + \frac{1}{2} \int_u^v g \mathbb{1}_{\{s^*(t)>0\}} dt + \frac{A}{2} = \frac{1}{2} \mathcal{E}_2^*|_{[u,v)} + \frac{1}{2} \mathcal{E}_3^*|_{[u,v)} \end{aligned}$$

where the third inequality is again due to the fact that the total idle duration in $[u, v)$ incurs a static energy A . \square

The above proofs of the claims complete the theorem proof. \square

References

- [1] Susanne Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010.
- [2] Susanne Albers and Antonios Antoniadis. Race to idle: new algorithms for speed scaling with a sleep state. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1266–1285, 2012.
- [3] S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1228–1241, 2012.
- [4] Evripidis Bampis, Christoph Dürr, Fadi Kacem, and Ioannis Milis. Speed scaling with power down scheduling for agreeable deadlines. *Sustainable Computing: Informatics and Systems*, 2(4):184–189, 2012.
- [5] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.
- [6] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 693–701, 2009.
- [7] Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, 2004.
- [8] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.
- [9] Ho-Leung Chan, Tak Wah Lam, and Rongbin Li. Tradeoff between energy and throughput for online deadline scheduling. In *Proc. 8th Workshop on Approximation and Online Algorithms*, pages 59–70, 2010.
- [10] Nikhil R. Devanur and Zhiyi Huang. Primal dual gives almost optimal energy efficient online algorithms. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms*, 2014.
- [11] Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proc. 44th ACM Symposium on Theory of Computing*, pages 137–144, 2012.
- [12] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Proc. 10th Workshop on Approximation and Online Algorithms*, pages 173–186, 2012.
- [13] Xin Han, Tak Wah Lam, Lap-Kei Lee, Isaac Kar-Keung To, and Prudence W. H. Wong. Deadline scheduling and power management for speed bounded processors. *Theor. Comput. Sci.*, 411(40-42):3587–3600, 2010.

- [14] Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. *FOCS 2014*, to appear.
- [15] Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *STOC*, 2014.
- [16] Sandy Irani, Sandeep K. Shukla, and Rajesh Gupta. Algorithms for power savings. *ACM Transactions on Algorithms*, 3(4), 2007.
- [17] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4): 617–643, 2000.
- [18] Peter Kling and Peter Pietrzyk. Profitable scheduling on multiple speed-scalable processors. In *Proc. 25th Symposium on Parallelism in Algorithms and Architectures*, 2013.
- [19] Kirk Pruhs and Clifford Stein. How to schedule when you have to buy your energy. In *APPROX-RANDOM*, pages 352–365, 2010.
- [20] Nguyen Kim Thang. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *Proc. 21st European Symposium on Algorithms*, pages 755–766, 2013.
- [21] F. Frances Yao, Alan J. Demers, and Scott Shenker. A scheduling model for reduced cpu energy. In *FOCS*, pages 374–382, 1995.

Appendix

Lemma 0 (Weak duality) *Consider a possibly non-convex optimization problem*

$$p^* := \min_x f_0(x) \quad : \quad f_i(x) \leq 0, \quad i = 1, \dots, m.$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ for $0 \leq i \leq m$. Let \mathcal{X} be the feasible set of x . Let $L : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ be the Lagrangian function

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x).$$

Define $d^* = \max_{\lambda \geq 0} \min_{x \in \mathcal{X}} L(x, \lambda)$ where $\lambda \geq 0$ means $\lambda \in \mathbb{R}_+^m$. Then $p^* \geq d^*$.

Proof We observe that, for every feasible $x \in \mathcal{X}$, and every $\lambda \geq 0$, $f_0(x)$ is bounded below by $L(x, \lambda)$:

$$\forall x \in \mathcal{X}, \forall \lambda \geq 0 : f_0(x) \geq L(x, \lambda)$$

Define a function $g : \mathbb{R}^m \rightarrow \mathbb{R}$ such that

$$g(\lambda) := \min_z L(z, \lambda) = \min_z f_0(z) + \sum_{i=1}^m \lambda_i f_i(z)$$

As g is defined as a point-wise minimum, it is a concave function.

We have, for any x and λ , $L(x, \lambda) \geq g(\lambda)$. Combining with the previous inequality, we get

$$\forall x \in \mathcal{X} : f_0(x) \geq g(\lambda)$$

Taking the minimum over x , we obtain $\forall \lambda \geq 0 : p^* \geq g(\lambda)$. Therefore,

$$p^* \geq \max_{\lambda \geq 0} g(\lambda) = d^*.$$

□